Inventaires systématiques et sémantiques du patrimoine et des musées

Travail de Bachelor réalisé en vue de l'obtention du Bachelor HES en Informatique de Gestion

par :

Loïc MICHOUD

Conseiller au travail de Bachelor :

Johann SIEVERING

Genève, le 19 février 2013

Haute École de Gestion de Genève (HEG-GE)

Informatique de Gestion

Déclaration

Ce travail de Bachelor est réalisé dans le cadre de l'examen final de la Haute école de gestion de Genève, en vue de l'obtention du titre de Bachelor of Science HES-SO en Informatique de gestion. L'étudiant accepte, le cas échéant, la clause de confidentialité. L'utilisation des conclusions et des recommandations formulées dans le travail de Bachelor, sans préjuger de leur valeur, n'engage ni la responsabilité de l'auteur, ni celle du conseiller au travail de Bachelor, du juré et de la HEG.

« J'atteste avoir réalisé seul le présent travail, sans avoir utilisé des sources autres que celles citées dans la bibliographie. »

Fait à Genève, le 19 février 2013

Loïc MICHOUD

Remerciements

Je tiens à remercier tout spécialement Monsieur **Johann SIEVERING** pour m'avoir mis ce sujet intéressant à disposition, pour ses conseils avisés et de m'avoir suivi durant toute la durée de mon travail.

Mademoiselle **Charlotte DELANNEE** pour m'avoir aidé à comprendre le fonctionnement des inventaires de musées et pour m'avoir fourni les informations nécessaires à cette compréhension.

Monsieur **Andréas SCHWEIZER** pour ses conseils et son aide dans la compréhension du système de l'API.

Finalement, Monsieur **Jean-Claude GENOUD** pour m'avoir fourni les informations concernant le fonctionnement et la structure des musées.

Résumé

Le but de ce projet est de donner une vue homogène d'un système d'informations muséal hétérogène.

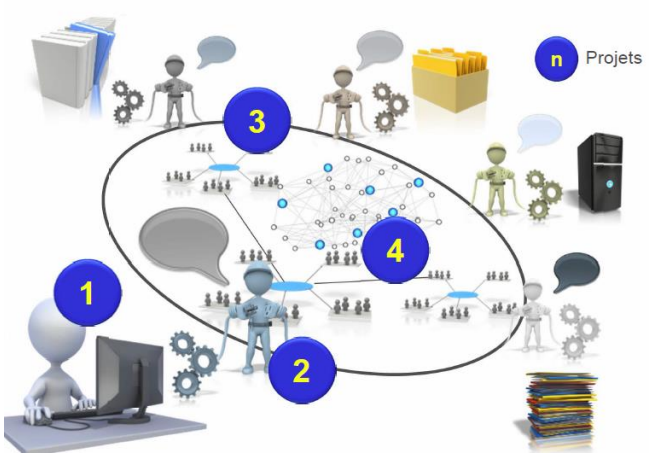
Ce qui permettra d'une part de faire des requêtes sur un ensemble de musées sans se préoccuper des aspects techniques sous-jacents et d'autre part de consolider les données réparties dans des sources n'ayant pas, *a priori*, de relations explicites.

Cette homogénéité doit être mise en place par une architecture particulière impactant :

- la technologie
  il faut pouvoir interroger des bases de données de diverses technologies ;
- les modèles des bases de données
  il faut pouvoir interroger plusieurs bases de données simultanément même si leurs structures de données ne sont pas identiques ;
- le métier
  il faut intégrer la façon dont la base a été pensée à son origine et son objectif local.

Figure 0

Présentation de l'architecture générale



Source : Johann Sievering

Ce mémoire fait partie d'un projet articulé en quatre volets et dont certaines parties ont été réalisées par plusieurs étudiants de la HEG. Les quatre volets de ce projet sont : les interfaces hommes-machines agiles, les connecteurs, la communication par agent et l'ontologie commune.

> Mon travail concerne la partie « interfaces hommes-machines agiles ».
>
> Cela implique l'utilisation de technologies web qui se chargera d'interpréter les résultats issus d'une recherche et de les présenter dynamiquement sur une page web.

Ce mémoire est constitué de cinq parties et d'une conclusion. La première partie présente l'état de l'art des systèmes de classification des collections des musées et particulièrement de Genève (API) et Lausanne (Muséris). La seconde partie décrit le mandat qui m'a été confié. La troisième partie analyse le domaine et les démarches que j'ai mis en œuvre. La quatrième partie s'étend sur les technologies utilisées dans le cadre du mandat. La cinquième partie détaille la réalisation du mandat et les étapes réalisées. La conclusion du mémoire présente une synthèse de l'ensemble du projet et une discussion sur mon expérience.

Table des matières

Déclaration i
Remerciements ii
Résumé iii
Table des matières v
Liste des Tableaux vii
Liste des Figures vii
Introduction 1
1. Présentation des différents organismes du projet 2
1.1 API 2
1.2 Social-IN^3 2
1.3 Muséris 3
2. État de l'art 4
2.1 Les outils de collections de musées 4
2.2 Le système de bases de données patrimoniales de Suisse romande 4
3. Mandat 7
4. Analyse du domaine 9
5. Réalisation du mandat 13
5.1 Réalisation de l'interface 13
5.2 Intégration 15
5.2.1 Explication des connexions 15
5.2.2 Intégration avec le connecteur 16
5.3 Environnement de développement 17
5.4 Choix du plugin de gestion des fichiers OWL 18
5.5 Choix du plugin de transformation de tableaux en « grid » 19
5.6 Prise en main de jOWL 19
5.7 Prise en main de jQuery 20
5.8 Liste des plugins utilisés 20
5.9 Prototype de l'interface 22
5.10 Le prototype de fichier OWL 24
5.11 Fonctionnement du code 26
6. Technologies 30
6.1 HTML 5 30
6.2 JavaScript 32
6.2.1 Jquery 33
6.3 RDF/OWL 34
6.3.1 RDF 34
6.3.2 RDFS 35
6.3.3 OWL 35
6.3.4 Logiciel d'édition d'ontologies « Protégé » 37

6.4 SPARQL 38
6.5 SPARQL-DL....38
Conclusion....40
Lexique 41
Bibliographie 42
Annexe 1 Aperçu de l'analyse du domaine....43
Annexe 2 Courriel 47
Annexe 3 Explication de l'utilisation d'AgroUML....49
Annexe 4 Installation d'Eclipse et du plugin Aptana 50
Annexe 5 Plugin permettant de transformer l'OWL en HTML 5 52
Annexe 6 Prototype de fichier OWL 55
Annexe 7 Code d'extraction du fichier OWL....62
Annexe 8 Fichier CSS de démonstration 65
Annexe 9 Fichier JavaScript de démonstration 68
Annexe 10 Fichier HTML de démonstration....69

Liste des Tableaux

Tableau 1 Planning hebdomadaire 7
Tableau 2 Planning du projet par thématiques 8
Tableau 3 Informations reçues à la suite du courriel 12
Tableau 4 Choix du plugin de gestion OWL 19
Tableau 5 Liste des plugins jQuery 21

Liste des Figures

Figure 1 Aperçu accueil de Muséris 5
Figure 2 Aperçu de la recherche de Muséris..... 6
Figure 3 Interface du logiciel AgroUML 9
Figure 4 Situation globale 10
Figure 5 Projet d'inventaire systématique..... 11
Figure 6 Interface dynamique (accueil)..... 13
Figure 7 Interface dynamique (exemple avec un fichier OWL) 14
Figure 8 Présentation de l'architecture générale 15
Figure 9 Prototype de l'interface (liste de résultats)..... 22
Figure 10 Prototype de l'interface (fiche détaillée)..... 23
Figure 11 Représentation de l'architecture du projet 26
Figure 12 Représentation UML de jOWL.retrive.js 26
Figure 13 Représentation du fichier jIntDynamic.js 28
Figure 14 Représentation du fichier jTransRequest.js 29
Figure 15 Représentation de l'ensemble des fichiers..... 29
Figure 16 Représentation des technologies HTML5..... 30
Figure 17 Présentation d'une page simple en HTML 5..... 31

Figure 18 Présentation des groupes fonctionnels HTML 5 32

Figure 19 Graphique représentant un exemple de mon fichier RDF ……. 34

Figure 20 Schéma représentant les couches du web sémantique..... 36

Figure 21 Interface de Protégé 4.2……. 37

Figure 22 AgroUML ajout d'arguments ……. 49

Figure 23 Fenêtre Eclipse d'ajout de plugin ……..... 50

Figure 24 GUI d'Eclipse avec le plugin Aptana ……..... 51

Introduction

Les interfaces hommes-machines agiles sont des interfaces qui ont la capacité à adapter la présentation des données au contenu voulu sans connaissances préalables des données sources. Par exemple, si l'on fait des recherches simultanées sur plusieurs sites, il peut arriver que les données retournées ne soient pas de même nature (un site retourne : une image, un texte, un lien Internet et un autre site retourne : une vidéo, un texte, une image qu'il faut présenter sur une même interface). Une interface standard a besoin de connaître à l'avance la nature et la structure des données qu'elle va recevoir, l'interface agile pourra s'adapter au contenu dynamiquement.

Les données reçues sont transmises dans le format RDF[1] (Resource Description Framework) et sont ensuite traitées localement pour être lues, interprétées et pour transformer le DOM[2] afin d'être présentées selon l'attente de l'utilisateur. L'interaction doit pouvoir être possible quelle que soit la source et la nature des données. C'est pourquoi les comportements doivent également être pris en charge par l'interface et le système d'informations. (Par exemple pour la gestion des actions). Cette interaction sera mise en place par le biais de fichiers JavaScript. Une gestion de l'affichage personnalisé doit être mise en place.

[1] http://fr.wikipedia.org/wiki/Resource_Description_Framework

[2] http://fr.wikipedia.org/wiki/Document_Object_Model

1. Présentation des différents organismes du projet

1.1 API

L'Association pour le Patrimoine Industriel (API) – fondée à Genève en 1976 a pour vocation de sauvegarder, conserver et valoriser les témoignages de la culture industrielle et technique d'importance cantonale, régionale ou nationale.

Reconnue d'utilité publique, elle œuvre à sensibiliser la population, les associations, les industriels ainsi que les autorités publiques et culturelles à la culture industrielle et technique.

La cohérence voulue par l'API l'a conduite à ériger quatre piliers, à partir desquels toutes les réflexions et toutes les pratiques sont possibles et parfois réalisées.

- **Patrimoine** (la conservation d'objets mobilier de la tradition des arts graphiques genevoise dans l'ancienne Usine de graisses industrielles Lambercier & Cie, construite en 1895)
- **Culture** (la transmission des savoir-faire de la tradition des arts graphiques)
- **Art** (la création contemporaine en couplant les techniques d'impression historique aux nouvelles technologies)
- **Social** (une plate-forme d'insertion sociale et professionnelle pour des demandeurs d'emplois et bénéficiaires de l'aide sociale, qui peuvent ainsi confronter leurs métiers pratiques)

Le directeur de l'API (depuis 1998) est Monsieur **Andréas Schweizer**.

1.2 Social-IN³

L'objectif de la coopérative social-IN3 est de mettre en relation différents acteurs et actrices qui souhaitent contribuer ensemble à l'émergence de produits ou de services innovants. Cette coopérative a comme ambition de développer un nouveau modèle de coopération dans le domaine scientifique, de la recherche, de l'invention et de la production industrielle. Elle s'inscrit dans l'économie sociale et solidaire pour développer une nouvelle forme de gouvernance et un cadre de développement pour la réalisation d'idées individuelles et collectives et la défense des intérêts économiques de ses membres.

1.3 Muséris

Muséris est une collection de bases de données de plusieurs musées lausannois mis en place par la ville de Lausanne et qui contient notamment (collections, bibliothèques, personnes, archives, cadres, documentation, restaurations et médiathèque) que l'on peut consulter indépendamment.

2. État de l'art

2.1 Les outils de collections de musées

Les outils de gestion de collections de musées sont nombreux, ils permettent une modularité, une personnalisation et un déploiement rapide, ils respectent les normes et les standards en vigueur et sont sécurisés. Par exemple : WebMuseo par aa-partners, Adlib Musée par Adlib, Flora Musée par Ever-team ou encore Micromusee par Mobydoc. En France, le bureau de la diffusion numérique des collections du service des musées de France et les musées participants ont créé un portail de collections : Joconde.[3]

Ces outils sont très complets et peuvent être utilisés par les musées s'ils le désirent. Mais ces outils ne permettent pas de mettre en relation des bases de données hétérogènes sans en modifier l'architecture.

2.2 Le système de bases de données patrimoniales de Suisse romande

Le système de bases de données patrimoniales de Suisse romande mis en place par la ville de Lausanne « Muséris » est décrit:

Sur le plan patrimonial et méthodologique :

- Un système cohérent, respectueux des normes et « multi métiers » ;
- Bibliothèques normées Marc;
- Archives normé ISAD-G;
- Appartenance à Europeana[4];
- Inter relation entre ces bases et entre les objets individualisés de ceux-ci;
- Ensemble de partenaires publics et privés (par exemple, la Ville de Lausanne, la Ville de Lancy).

[3] Liste des prestataires de logiciels de gestion de collection de musées : http://www.culture.gouv.fr/documentation/joconde/fr/partenaires/AIDEMUSEES/societe-info.htm

[4] http://fr.wikipedia.org/wiki/Europeana

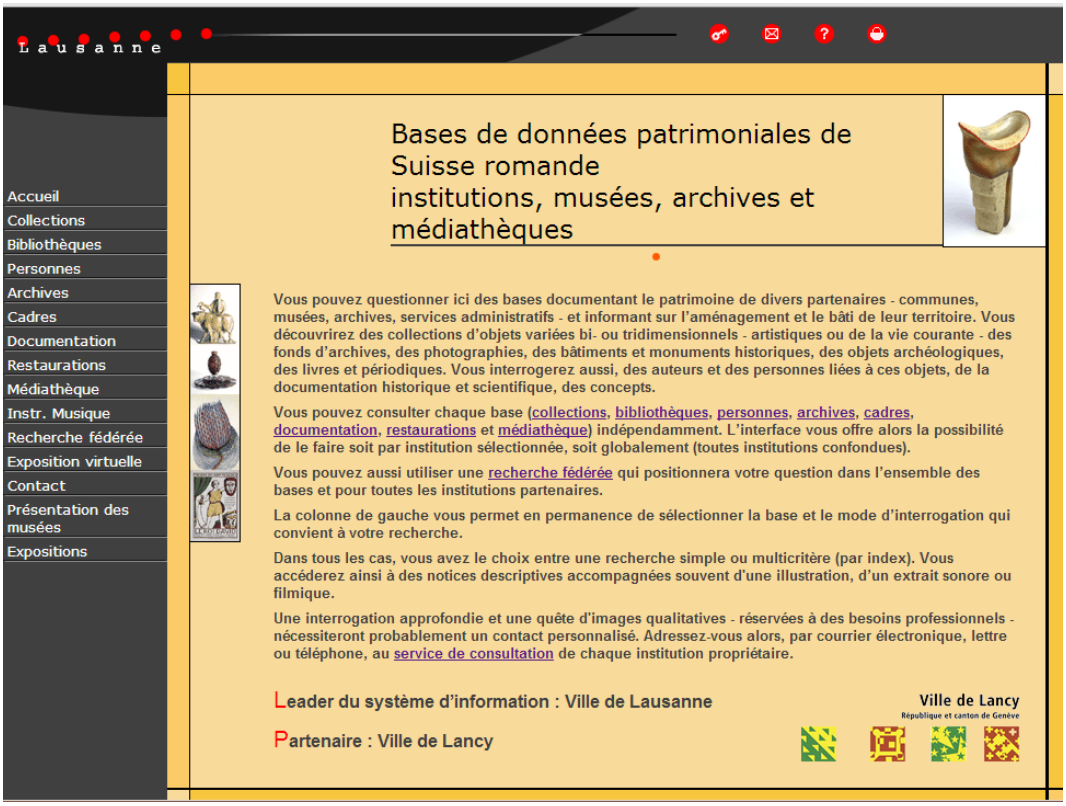
Sur le plan technique et pratique :

- Une gestion des droits personnalisés par partenaire
- Un accès à distance par page web en consultation et catalogage
- Un niveau de sécurité élevé

Ce système est donc assez complet, mais il ne permet pas de mettre directement en commun des bases de données de plusieurs musées.

Figure 1

Aperçu de l'acceuil de Muséris

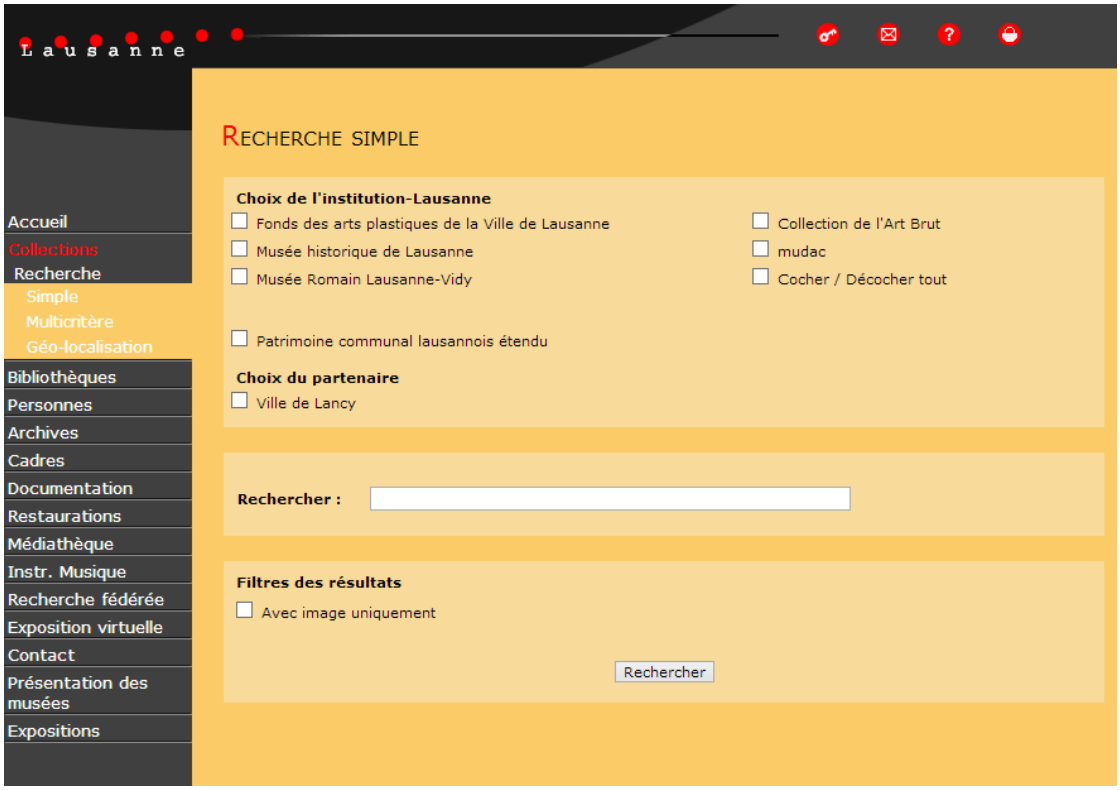


Source : https://museris.lausanne.ch/

On peut voir ci-dessus un aperçu de l'accueil du système de bases de données patrimoniales de Suisse de son nom « Muséris ».

Figure 2

Aperçu de la recherche de Muséris



Source : https://museris.lausanne.ch/

La figure ci-dessus illustre le système de recherches mis en place par « Muséris ».

Ce formulaire fonctionne de la manière suivante : après avoir sélectionné les musées cibles, la requête est saisie dans la zone de texte et ensuite confirmée.

Cependant, ces outils de collections de musées ne permettent pas de relier des bases de données entre elles sans que les musées aient à effectuer des changements dans leur infrastructure. Une interface dynamique est une étape indispensable à la réalisation d'un tel système, car on ne connaît pas a priori le type de données que l'on va recevoir et c'est pourquoi il faut les gérer dynamiquement.

3. Mandat

Le mandat qui m'a été confié est le suivant :

> « Étude d'une interface homme-machine agile qui permet d'afficher dynamiquement les résultats des recherches ».

Mon travail de Bachelor a été fixé à 23 semaines à 40% d'activité (16h) réparties selon le planning ci-dessous.

Tableau 1

Planning hebdomadaire

Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
Bachelor(4h)	Bachelor(4h)	TP	Bachelor(4h)	Bachelor(4h)		
Chinois	GL/AuditBDD	Wifi	TP GL / BIA	Datamining		

Date de début	10.sept.12	**Date de fin**	18.févr.13
Date de soutenance	22.févr.13		

Ces 23 semaines sont structurées selon les objectifs suivants :

Le 10 septembre 2012 a eu lieu la présentation du projet et de l'équipe à l'API. La première semaine a été consacrée à l'apprentissage du fonctionnement de l'API et à la maîtrise du projet. Les semaines 2, 3 et 4 ont été consacrées à la mise en place des « use-cases[5] » et à la modélisation du système. La semaine 5 a été consacrée à la conception d'un diagramme filaire[6]. Les semaines 6 et 7 ont été utilisées pour demander les informations nécessaires à la création de l'analyse du domaine et à l'écriture du mémoire. La semaine 8 a servi à la création de l'analyse du domaine. Les semaines 9 et 10 ont servi à la compréhension de RDF et OWL et à la mise en place d'un prototype de fichier OWL. Les semaines 11 à 14 ont été consacrées au choix d'un plugin pour interpréter les fichiers RDF/OWL et à la création d'un plugin permettant la transformation du fichier OWL en HTML 5. La semaine 15 a été consacrée à la présentation des sujets et à l'élaboration du plan d'actions pour l'intégration. Les semaines 16 et 17 ont dû être

[5] Un diagramme de Use Case permet d'identifier les fonctionnalités que doit fournir le système.

[6] Un diagramme filaire est une maquette de fidélité moyenne permettant de représenter une interface. (voir Figure 16 page 30)

utilisées pour les projets de datamining et de business intelligence qui étaient considérés comme des examens. Les semaines 18 et 19 ont été consacrées aux examens. Les semaines 20 et 21 ont été utilisées pour la rédaction du mémoire. Finalement, les semaines 22 et 23 ont été utilisées pour la finalisation du code et à la correction du mémoire.

Tableau 2

Planning du projet par thématiques

Dates	Thématiques
10.09.12 - 16.09.12	Apprentissage du fonctionnement de l'API et de la maîtrise du projet
17.09.12 - 07.10.12	Mise en place des Use Cases et modélisation du système
08.10.12 - 14.10.12	Conception des diagrammes filaires
15.10.12 - 28.10.12	Recueil d'informations pour l'analyse du domaine et l'écriture du mémoire
29.10.12 - 04.11.12	Création de l'analyse du domaine
05.11.12 - 18.11.12	Compréhension de RDF et OWL et mise en place d'un prototype de fichier OWL
19.11.12 - 16.12.12	Choix du plugin d'interprétation RDF/OWL et création du plugin permettant la transformation d'un fichier OWL en HTML5
17.12.12 - 23.12.12	Présentation des sujets et élaboration du plan d'action pour l'intégration
24.12.12 - 06.01.13	Projet de Datamining et Business Intelligence
07.01.13 - 20.01.13	Examens HEG
21.01.13 - 03.02.13	Rédaction du mémoire
04.02.13 - 17.02.13	Finalisation du code et correction du mémoire

4. Analyse du domaine

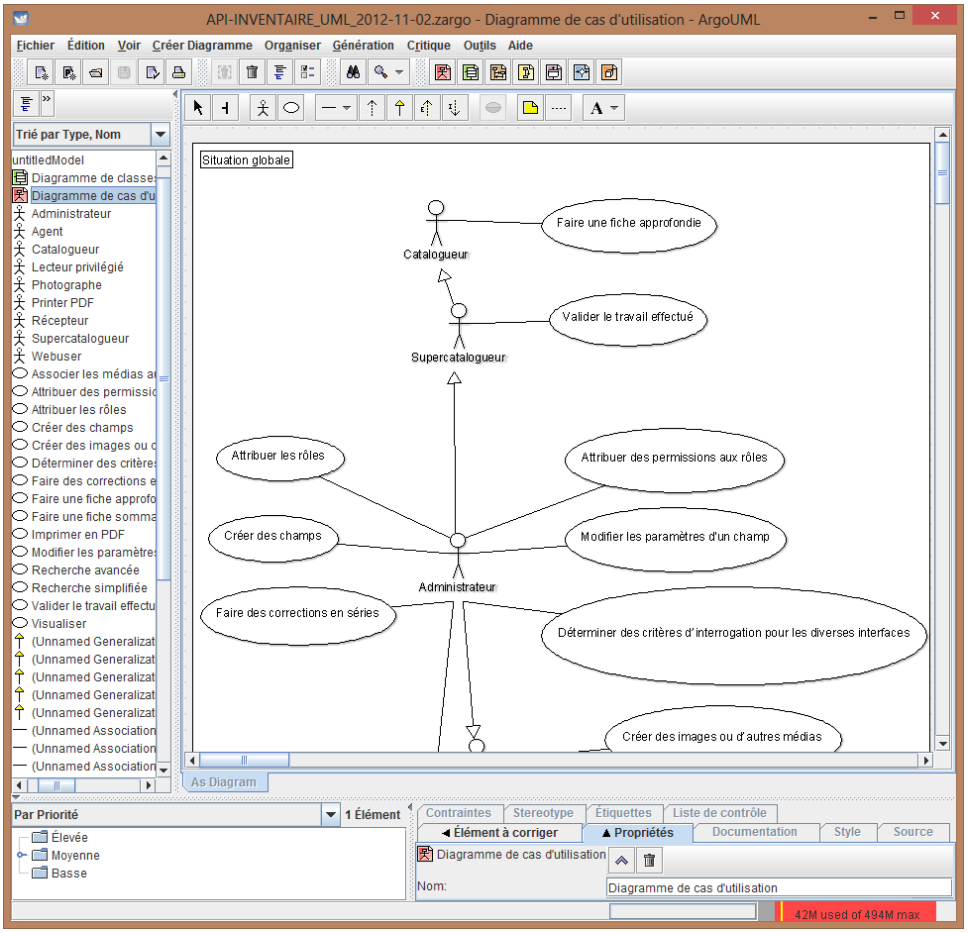
L'analyse du domaine s'est faite en plusieurs phases.

La première phase a été de comprendre le fonctionnement du système. Pour cela, il a été nécessaire de collecter des informations aux personnes s'occupant des systèmes pour en connaître les acteurs et leurs actions.

La deuxième phase a consisté à concevoir des use-cases qui correspondent au besoin du système. Pour ce faire j'ai utilisé le logiciel AgroUML[7] dont voici un aperçu :

Figure 3

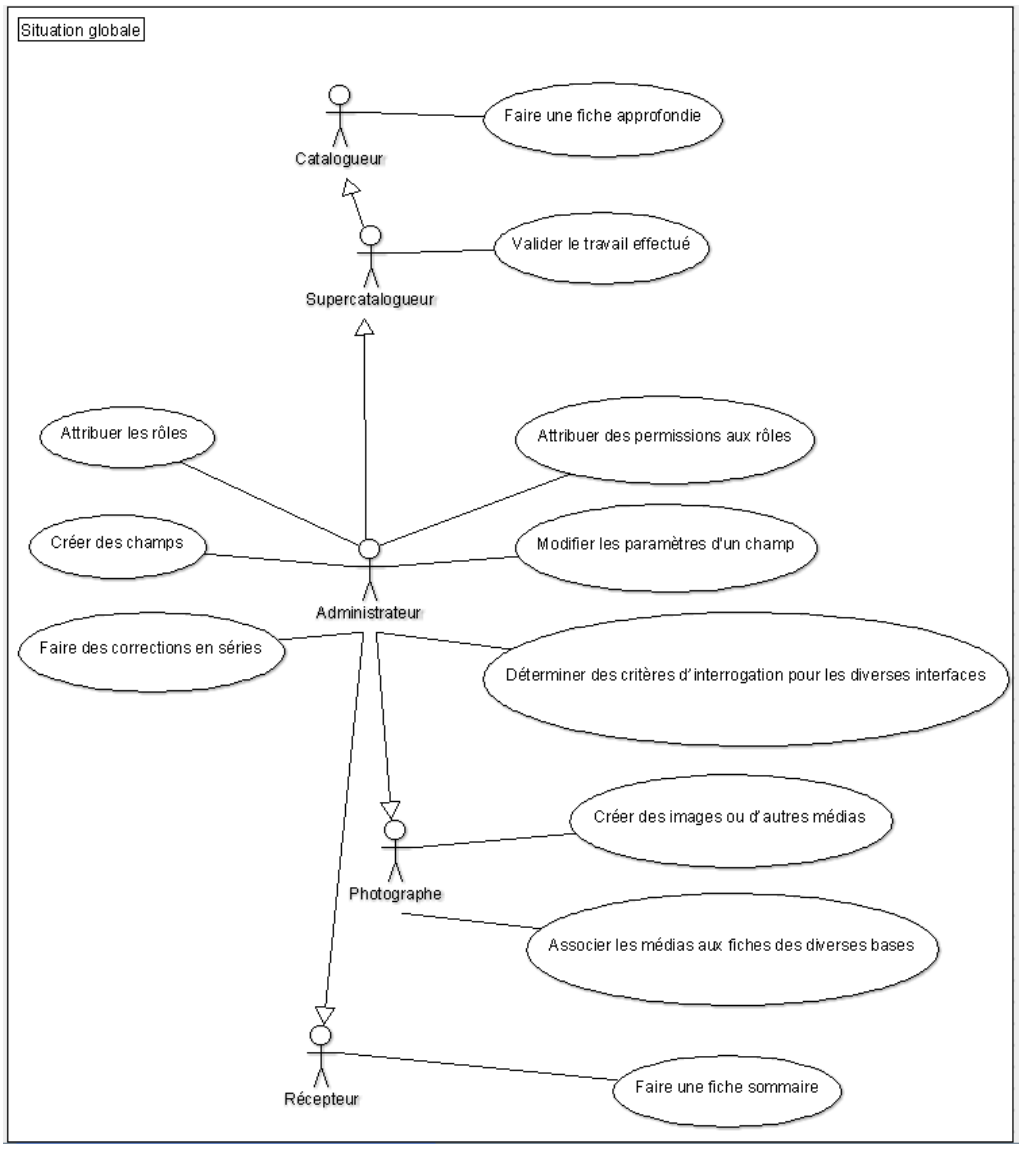
Interface du logiciel AgroUML



[7] Site web de l'auteur : http://argouml.tigris.org/

Figure 4

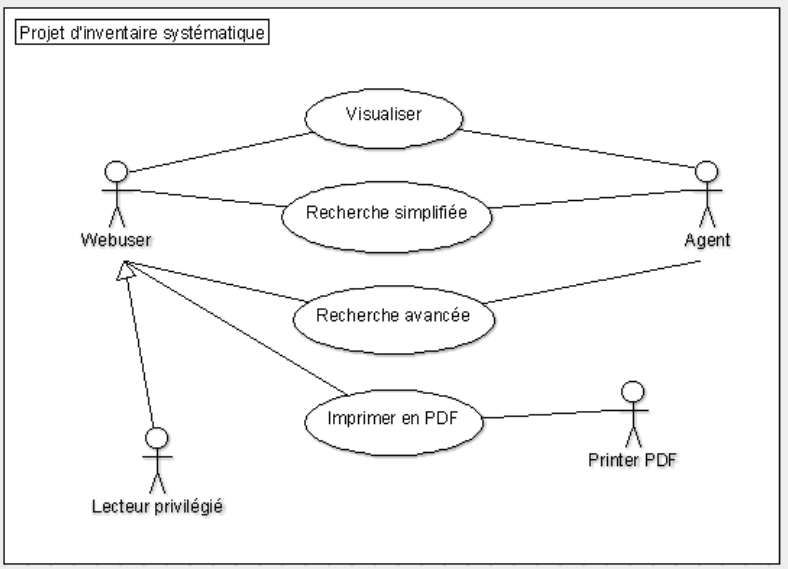
Situation globale



La figure ci-dessus montre que l'on a cinq acteurs distincts qui ont des tâches bien particulières bien qu'un acteur puisse effectuer les tâches d'un autre acteur s'il hérite de ses fonctions grâce à la flèche d'héritage. Cela est la situation globale.

Figure 5

Projet d'inventaire systématique



Cette figure nous montre le domaine du mandat. On peut voir que l'utilisateur de base est l'internaute qui n'est pas identifié par le système contrairement au lecteur privilégié qui possède des droits qui lui sont propres. L'acteur secondaire « Agent » est celui qui s'occupera de traiter les recherches et de rendre les résultats.

La troisième phase a consisté à mettre en place les détails des acteurs et leurs rôles (nom, qualité, use-case, fonction, compétences, droits, but, résultat, collaborations, importance).[8] Pour cela, il a fallu demander de plus amples informations aux personnes compétentes dans le domaine.

Dans ce but j'ai donc écrit un courriel dont le contenu est présent en annexe[9]. À la suite de mon courriel, j'ai reçu toutes les informations dont j'avais besoin pour compléter mon analyse de domaine.

Une explication de l'utilisation d'AgroUML est présente dans les annexes[10].

Ci-dessous, un tableau synthétique des informations reçues après le courriel.

[8] Voir Annexe 1 (toutes les colonnes ne sont pas présentes, il manque les colonnes qualité, cas d'utilisation, compétences et collaborations. Le document entier se trouve sur le CD).

[9] Voir Annexe 2.

[10] Voir Annexe 3.

Tableau 3

Informations reçues à la suite du courriel

Informations reçus à la suite du courriel	
Mr. Schweizer	**Mr. Genoud**
Rôles	Rôles
	Droits
	Localisation
	Fonctions
	Nombre de personnes
	Compétences
	Document sur le système d'information documentaire des musées lausannois

5. Réalisation du mandat

5.1 Réalisation de l'interface

La réalisation de l'interface a été effectuée en HTML 5 et CSS 3. L'utilisation de la bibliothèque JavaScript « jQuery » a été utilisée pour sa facilité, sa puissance, son nombre important de plugins et une connaissance du fait que je l'ai déjà utilisée à plusieurs reprises personnellement.

Une des nouveautés de CSS 3 est la possibilité d'ajouter des dégradés, des arrondis, des ombres et d'ajouter des polices grâce à @face-font.

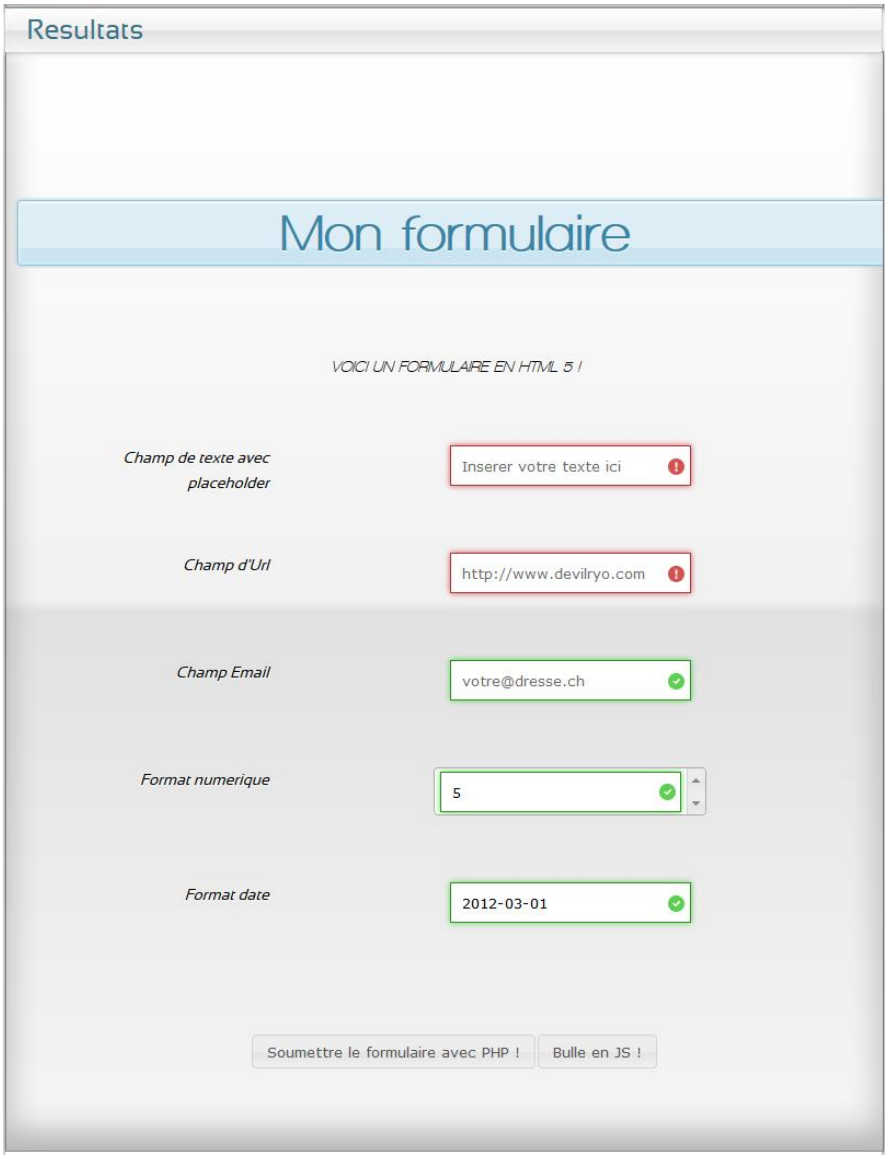
Ci-dessous, un aperçu de l'interface dans son ensemble.

Figure 6

Interface dynamique (accueil)

Figure 7

Interface dynamique (exemple avec un fichier OWL)



Ci-dessus, on peut visualiser ce que donne l'interface créée dynamiquement à partir du fichier OWL.[11]

Ce formulaire a été généré dynamiquement, il comporte un fichier CSS [12] pour sa mise en page ainsi qu'un fichier JavaScript [13] pour les comportements des boutons.

[11] Fichier OWL que vous pouvez trouver en Annexe 6.

[12] Le code du fichier CSS utilisé pour la démonstration se trouve en Annexe 8.

[13] Le code du fichier JavaScript du formulaire se trouve en Annexe 9.

5.2 *Intégration*

5.2.1 Explication des connexions

L'interface agile envoie une requête de recherche à un agent qui l'interprète. L'agent demande les informations au connecteur qui les lui rend dans le format OWL.

L'agent retourne les résultats à l'interface agile sous forme d'un fichier OWL qui contient des objets «composants» HTML.

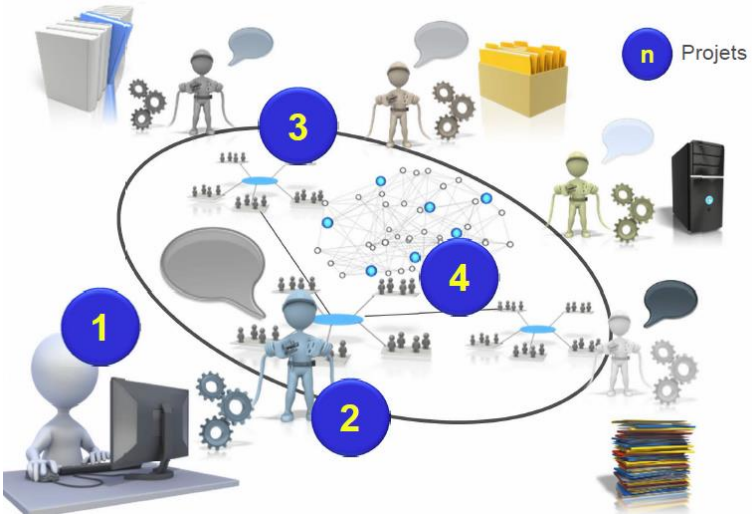
Ce fichier OWL est composé de la couche de données qui contient les «composants» HTML, une couche de présentation qui peut contenir un ou plusieurs URI de fichiers CSS et une couche de comportement qui peut contenir un ou plusieurs URI de fichiers JavaScript.

Ce fichier OWL est ensuite lu et interprété par l'interface agile qui crée ensuite les composants HTML.

Synoptique de l'architecture :

Figure 8

Présentation de l'architecture générale



Source : Johann Sievering

Le point 1 représente les IHM agiles et donc la partie pour laquelle j'ai été mandaté. Le point 4 représente le réseau d'agents qui vont discuter entre eux pour faire les meilleures recherches possibles. Grâce au point 3, on peut se rendre compte que peu importe la technologie ou le format des bases de données à interroger les connecteurs pourront comprendre et se faire comprendre du système. Le point 2 montre la connexion entre le connecteur et l'IHM agile, car c'est lui qui permettra de recevoir les résultats des recherches.

5.2.2 Intégration avec le connecteur

L'intégration avec le connecteur va consister à s'assurer que toutes les informations requises pour la création de l'interface dynamique soient implémentées dans le fichier de résultats afin que le connecteur puisse transmettre à l'interface dynamique.

Lorsque toutes ces informations sont incluses dans le fichier, il faut traiter les données pour les transformer dans le format compatible à leurs implémentations sous la forme de composants HTML.

Pour ce faire, un lien vers un fichier « Template » est nécessaire. Ce fichier sert à savoir comment le musée veut afficher ses données par exemple sous un format XML. On aurait :

```
<?xml version="1.0" encoding="UTF-8"?>
<template>
      <title src="Machine" />
      <footer src="Groupe" />
</template>
```

De plus, la déclaration de la source du résultat est nécessaire pour ne pas mélanger les éléments et pour pouvoir facilement leurs donner un identificateur unique. Celui-ci peut être par exemple : « Lausanne01 ».

Les autres informations importantes nécessaires sont principalement le type de données et la longueur de la donnée. Il faut aussi avoir, si le musée l'a mis en place, l'URI du ou des fichiers de présentation pour un affichage spécifique ou l'URI du ou des fichiers de comportement pour permettre la mise en place d'un comportement spécifique, comme par exemple un bouton. Une information importante, mais pas écrite explicitement est le nombre de résultats retournés. En effet, s'il n'y a pas de données cela ne sert à rien de traiter le fichier, s'il n'y a qu'un seul résultat cela implique qu'il faudra afficher une fiche détaillée tandis que dans le cas où il y aurait plus d'un résultat cela implique un affichage en forme de tableau.

Une fois toutes ces informations transformées dans le format OWL de composant, le plugin qui s'occupe du traitement de fichier OWL et le plugin de transformation en HTML 5, font leur travail et l'interface s'affiche.

5.3 Environnement de développement

Dans un souci de réutilisation et de compatibilité avec les logiciels déjà utilisés localement, l'environnement de développement choisi est Eclipse avec les plugins nécessaires à la programmation: JavaScript, jQuery, CSS et HTML 5.

Le plugin utilisé pour notre développement est Aptana. Ce plugin produit par Appcelerator[14], est disponible en application autonome et en version plugin pour Eclipse. Aptana supporte les dernières spécifications technologiques pour le développement web comme HTML5, CSS3, JavaScript, Ruby, Rails, PHP et Python.

Un autre logiciel utile dans le cadre du développement est Notepad++[15]. Cet éditeur de code source prend en charge plusieurs langages de programmation et permet une édition facile des fichiers et une coloration des balises. Il est sous licence GPL.

Une explication de l'installation d'Eclipse ainsi que du plugin Aptana se trouve en annexe.[16]

[14] Site internet d'Aptana : http://www.aptana.com

[15] Site officiel de Notepad++ : http://notepad-plus-plus.org/fr/

[16] Voir annexe 4.

5.4 *Choix du plugin de gestion des fichiers OWL*

J'ai trouvé plusieurs plugins permettant la gestion et l'interprétation de fichiers RDF/OWL. Par la suite, j'ai fait une matrice de préférence pour trouver le plugin qui conviendrait le mieux.

Tableau 4

Choix du plugin de gestion OWL

ALTERNATIVES	RDF/OWL Coef. : 1	DOCUMENTATION Coef. : 0.5	LICENCE Coef. : 1	JAVASCRIPT Coef. : 1	SPARQL DSL Coef. : 0.8	FICHIER EXTERNE Coef. : 1
jOWL	1	1	1	1	1	1
jRDF	1	2	1	0	1	1
Raptor RDF	1	2	1	0	0	1
rdfQuery	1	1	1	1	1	0

Sur le plan de la gestion des fichiers RDF/OWL tous les plugins permettaient cette gestion. La documentation était suffisante pour jOWL et rdfQuery, mais n'était pas très explicite et assez difficile à comprendre. Par contre, la documentation pour jRDF et Raptor RDF était très complète et facile à prendre en main. Au niveau de la licence, tous les plugins autorisaient une utilisation libre. Malheureusement, jRDF et Raptor RDF n'utilisent pas JavaScript mais respectivement Java et C++. Pour l'utilisation de requêtes SPARQL, tous les plugins sauf Raptor RDF permettaient de rechercher dans les fichiers OWL par SPARQL. Pour ce qui est de la prise en charge de fichiers OWL, RDF externe, rdfQuery est le seul plugin ne le permettant pas.

En conclusion, les deux plugins qui étaient en course étaient jOWL et rdfQuery car tous deux utilisaient JavaScript. Par contre, pour la prise en charge de fichiers externes et aussi une meilleure prise en main j'ai choisi le plugin jOWL.

5.5 Choix du plugin de transformation de tableaux en « grid »

Plusieurs plugins permettent la transformation de tableaux en « grid ». Après avoir recherché sur Google les différents plugins, j'ai trouvé une liste de plugins qui fait mention de ces plugins avec leurs informations.[17]

Dans ces plugins, quatre ont été particulièrement comparés.

- SlickGrid
- jqGrid
- DataTables
- Flexigrid

Flexigrid a tout d'abord été testé car il convenait à l'utilisation voulut mais la fonction de tri des colonnes ne fonctionne que si on dispose d'une base de donnée car le plugin ne le gère pas. jqGrid permet un tri des colonnes mais contrairement à Fexigrid il ne permet pas de déplacer les colonnes. SlickGrid contient toutes les fonctions voulues mais il ne transforme pas un tableau. DataTables transforme du DOM, des structures tabulaires et dispose de toutes les fonctions désirées, c'est-à-dire : tri sur les colonnes, recherche, affichage paginé et déplacement des colonnes. Le plugin qui a été retenu est donc DataTables.

5.6 Prise en main de jOWL

La prise en main de jOWL a été assez compliquée au début, mais lorsque j'ai compris le fonctionnement des requêtes SPARQL intégrées dans le plugin pour faire les recherches cela a tout de suite été plus facile. jOWL commence par la méthode load() prenant comme argument le nom du fichier à traiter. Grâce à l'appel de requêtes SPARQL, il suffit de faire une nouvelle instance de l'objet SPARQL_DL() qui prend comme argument une requête SPARQL de type : "Thing(?i), Type(?i, ?C)" ce qui permet de prendre tous les individus et leurs classes respectives.[18]

Par la suite, il a fallu faire des boucles pour parcourir les individus et leur classe, faire d'autres requêtes SPARQL pour connaître leur sous-classe et leurs propriétés ainsi qu'instancier la classe qui s'occupe de transformer en HTML 5 pour implémenter l'interface agile en HTML 5.[19]

---

[17] Liste des plugins : http://wiki.jqueryui.com/w/page/35122722/Grid-OtherGrids

[18] Voir annexe 7.

[19] Voir annexe 5.

5.7 Prise en main de jQuery

La prise en main de jQuery a été assez facile, car je l'ai déjà beaucoup utilisé personnellement. Pour la création de nouvelles classes, jQuery n'était pas l'idéal, car il n'était pas orienté objet, mais j'ai découvert un plugin qui donnait la possibilité d'écrire de nouvelles classes dans une vision objet. Ce plugin se nomme jQuery.class[20] et convenait parfaitement pour l'utilisation dont j'avais besoin.

5.8 Liste des plugins utilisés

Dans le cadre du projet, j'ai été amené à utiliser plusieurs plugins jQuery que ce soit pour un souci d'esthétique ou pour des fonctionnalités supplémentaires utiles. Le tableau ci-dessous décrit les plugins utilisés, leurs licences (quand cela était possible à trouver) et les buts dans lesquels ils sont utilisés.

[20] Site web : http://javascriptmvc.com/docs.html#!jQuery.Class

Tableau 5

Liste des plugins jQuery

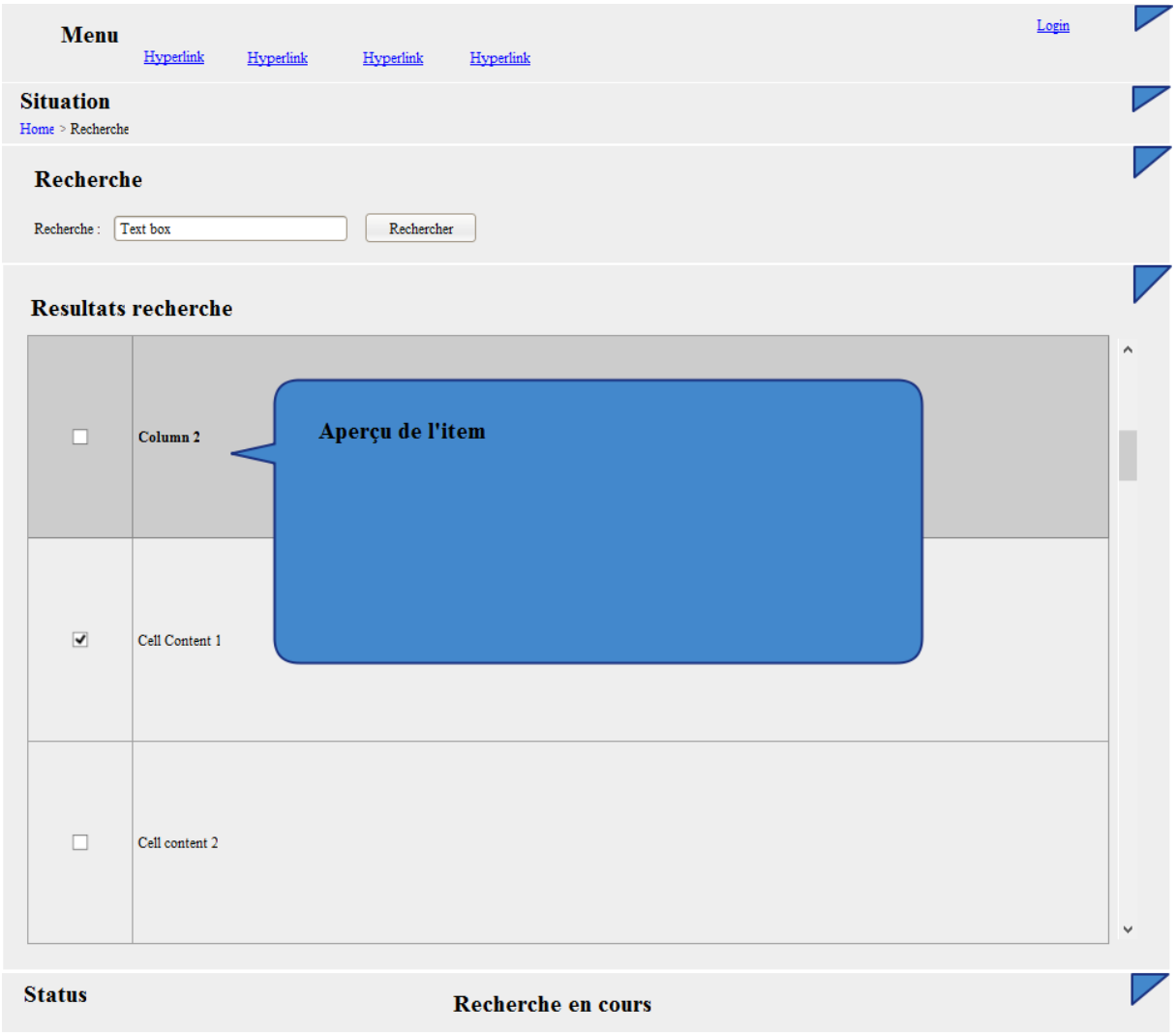
NOM PLUGIN	LICENCE	UTILISATION	PLATEFORME	VERSION	AUTEUR	SITE WEB
jQuery	MIT License	Librairie javascript	Javascript	1.8.3	The jQuery Foundation	http://jquery.com/
jQuery UI	MIT License	Addon de jQuery pour les UI	jQuery	1.9.2	The jQuery Foundation	http://jqueryui.com/
Modernizr	MIT License	Permet de détecter les fonctions HTML5 et CSS3 que le navigateur peut utiliser	Javascript	2.6.2	Modernizr	http://modernizr.com/
bPopup	N/A	Permet d'afficher une modalbox	jQuery	0.8	Bjoern Klinggaard	http://dinbror.dk/bpopup/
jQuery Class	N/A	Permet de créer des classes objets avec jQuery	jQuery	N/A	Javascript MVC	http://javascriptmvc.com/docs.html#!jQuery.Class
DataTables	GPL v2 license	Permet de transformer un tableau en grid	jQuery	1.9.4	Sprymedia	http://www.datatables.net
jOWL	MIT License	Parcourir des fichiers RDF/OWL et les parser	jQuery	1.0	David Decraene	http://jowl.ontologyonline.org/
Browser Detection	N/A	Permet de détecter quel navigateur l'utilisateur emploie	jQuery	1.1	Devslide	http://www.devslide.com/labs/browser-detection

5.9 *Prototype de l'interface*

Le prototype de l'interface a été effectué avec l'aide du logiciel Pencil[21] qui est disponible en composant autonome pour Linux, Windows et Mac ou comme extension Firefox. Ce logiciel a été facile à prendre en main et son utilisation est intuitive.

Figure 9

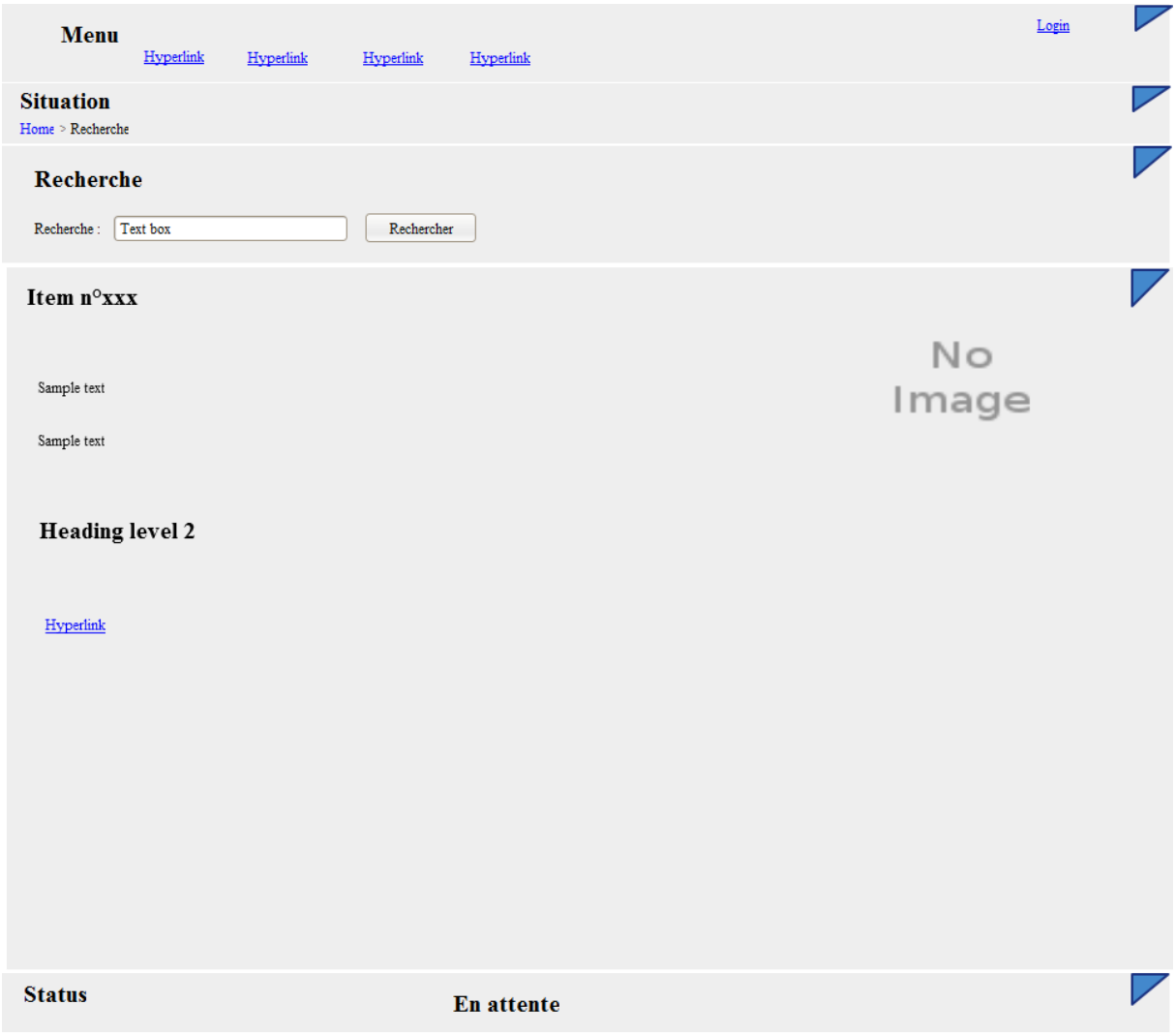
Prototype de l'interface (liste de résultats)



[21] http://pencil.evolus.vn/

Figure 10

Prototype de l'interface (fiche détaillée)



On peut voir sur les images du prototype que les sections sont horizontales. Cela permet un effet de type « collapse » en jQuery, ce qui permet de réduire les sections d'un simple clic et de les ragrandir de la même façon. Cet arrangement sert aussi à pouvoir adapter le contenu à plusieurs résolutions d'écrans comme des écrans de smartphones ou tablettes graphiques car il suffit de réduire la largeur de toutes les sections.

5.10 Le prototype de fichier OWL

Il a été nécessaire de créer un fichier OWL de test pour pouvoir vérifier le plugin jOWL et vérifier que l'implémentation de l'OWL en HTML 5 était correcte. Pour ce faire, il a été nécessaire que je me documente sur les technologies RDF et OWL et que je développe une ontologie de test permettant de gérer les composants HTML. Le fichier OWL a été décomposé en trois couches.

La première couche est la couche de présentation qui contient les URI (lien qui permet d'identifier une ressource) du ou des fichiers CSS (.css). Pour cela, a été créée une classe qui se nomme « link » pour faire le rapport avec la balise HTML <link> qui sert à ajouter un fichier CSS.

La seconde couche est la couche de comportement qui contient les URI du ou des fichiers JavaScript (.js). De la même manière que pour la couche présentation, une classe script a été créée qui a pour but de faire le lien avec la balise HTML <script> servant à ajouter un fichier JavaScript.

Pour terminer, la couche de données contient toutes les données reçues par rapport à notre recherche. Cette couche définit toutes les balises utiles à notre application comme form pour les formulaires ou encore div pour les paragraphes.

Il a fallu ensuite définir les propriétés qui seront utilisées dans la définition de nos individus comme par exemple la propriété type qui permet de définir de quel type est une balise <input> ou bien encore la propriété href qui permet d'ajouter un lien à une balise <a>.

La définition des individus se fait alors de la manière suivante :

Définition d'un label :

```
<Label rdf:about="#lText">
       <rdf:type rdf:resource="#Label"/>

      <form>monFormulaire</form>
      <for>iText</for>
      <inside>pText</inside>
      <text>Champ de texte avec placeholder</text>
</Label>
```

On a donc un label qui fait partie du formulaire avec l'identifiant « monFormulaire » qui est lié avec le champ de texte qui a l'identifiant « iText ». Celui-ci se trouve à l'intérieur de la balise pText qui est de type <p> et affiche « Champ de texte avec placeholder ».

Définition d'un champ texte :

```
<Input rdf:about="#iText">
       <rdf:type rdf:resource="#Input"/>

       <id>iText</id>
       <name>monText</name>
       <form>monFormulaire</form>
       <type>text</type>
       <inside>pText</inside>
       <placeholder>Insérer votre texte ici</placeholder>
       <required>required</required>
</Input>
```

Le champ texte est une balise « input » qui a pour type « text », a comme identifiant « iText » et fait partie du formulaire « monFormulaire ». Celui-ci se trouve à l'intérieur de la balise <p> qui a pour identifiant « pText ». Il contient un message d'aide « Insérer votre texte ici » et est requis pour envoyer le formulaire.

Ce fichier est ensuite analysé à l'aide du plugin jOWL pour pouvoir interpréter les couches, les balises et les propriétés. Cela est fait avec le fichier jOWL.retrive.js qui utilise jOWL.[22] Il va ensuite faire appel à la classe qui s'occupe de transformer ces informations en interface grâce à jQuery qui permet d'ajouter du code HTML à la volée.

Le fichier prenant en charge ce travail se nomme jIntDynamic.js [23] et est composé d'une super classe « IntDynamic.Layer » et d'une sous classe par couche ce qui donne donc trois sous classes qui portent le nom de leur couche. La couche présentation se nomme « IntDynamic.PresentationLayer », la couche comportement « IntDynamic.ComportementLayer » et pour finir la couche de données se nomme « IntDynamic.DataLayer ». La couche comportement et présentation ayant le même comportement font seulement appel à la fonction de leur super classe qui est le comportement générique, mais pour anticiper l'avenir, elles ont été séparées. La classe de données, quant à elle, implémente un comportement différent.

[22] Le code de ce fichier est en annexe 7.

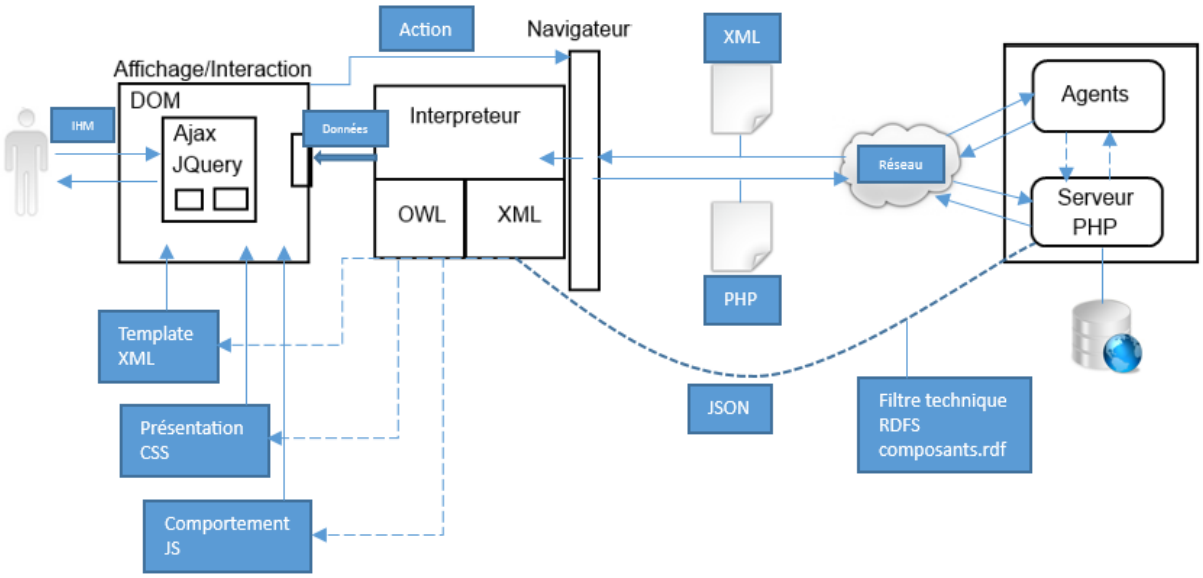
[23] Le code de ce fichier est en annexe 5.

5.11 Fonctionnement du code

L'architecture générale du projet :

Figure 11

Représentation de l'architecture du projet



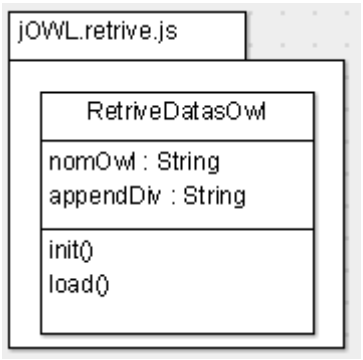
Le traitement des fichiers OWL et sa transformation en HTML a été implémenté dans deux fichiers distincts.

Le premier fichier jOWL.retrive.js contient une classe. Cette classe « RetriveDatasOwl » possède deux méthodes. La première méthode « init(nom, id) » se lance à l'instanciation de la classe et prend comme argument le nom du fichier OWL et l'identifiant de la balise à l'endroit où l'on veut que l'interface dynamique s'affiche. La seconde méthode « load() » traite le fichier OWL chargé.

```
new RetriveDatasOwl("data/monFormulaire.owl",
"interfaceDynamic").load();
```

Figure 12

Représentation UML de jOWL.retrive.js



Le second fichier jIntDynamic.js contient un espace de nommage « IntDynamic », une super classe et trois sous-classes. La super classe IntDynamic.Layer possède deux formes possibles, une sans argument et une avec le nom de la balise et l'identifiant de l'élément dans lequel on veut l'afficher. Elle possède également une méthode options() qui va permettre d'ajouter tous les noms d'attributs ainsi que leur valeur. Sa deuxième méthode add() permet d'ajouter un élément ainsi que ses attributs dans le div voulu. Elle implémente le comportement le plus commun à ses sous-classes.

Les sous-classes sont nommées en fonction de leur couche respectivement :

- IntDynamic.PresentationLayer
- IntDynamic.ComportementLayer
- IntDynamic.DataLayer

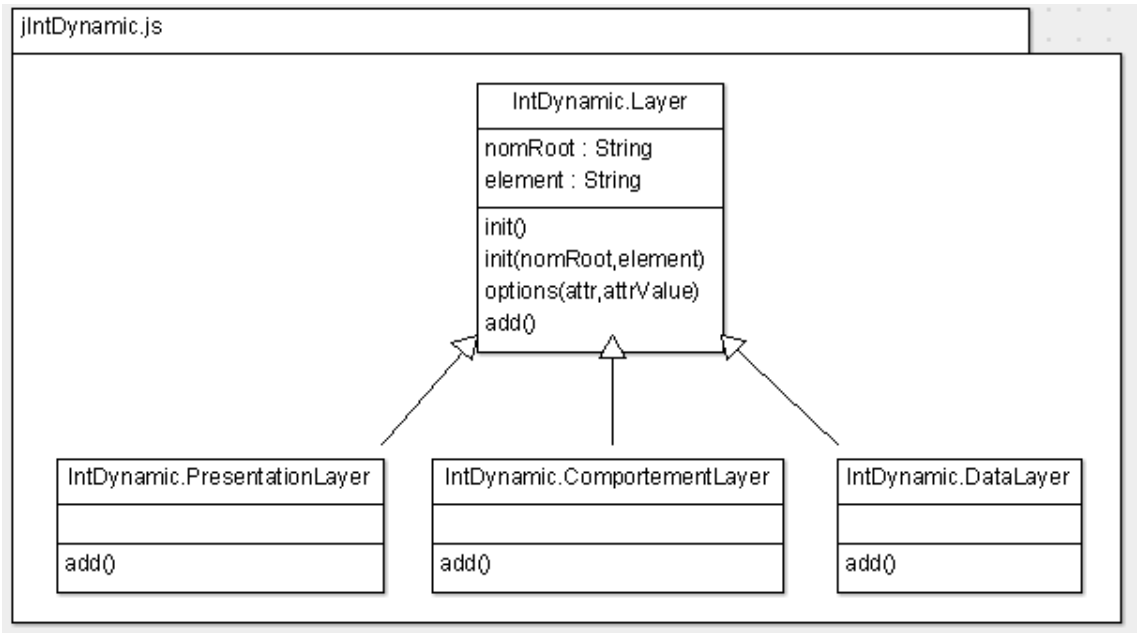
Les sous-classes ont un comportement commun et utilisent la méthode add() héritée de leur super classe. La sous-classe de données, quant à elle, implémente une méthode add() propre à son comportement. Voici un aperçu d'une instanciation de la super classe suivie d'une instanciation de la sous-classe IntDynamic.DataLayer :

```
component = new IntDynamic.Layer(this.constructor.maClasse,
appendHead);
component = new IntDynamic.DataLayer(this.constructor.maClasse,
appendDiv);
```

Une modélisation du fichier jIntDynamic.js permettant de visualiser sa super classe ainsi que ses sous-classes est représentée ci-dessous :

Figure 13

Représentation du fichier jIntDynamic.js



Le code pour traiter les fichiers XML reçus a été implémenté dans un fichier « jTransRequest.js » comportant six fonctions. C'est le fichier qui sera appelé quand une réponse à une recherche sera rendue. Voici un exemple d'appel :

```
$.ajax( {
    type: "GET",
    url: "./data/resultList.xml",
    dataType: "xml",
    success: function(xml) {
        monOwl = new
              IntDynamic.TransformRequest("interfaceDynamic");
        monOwl.processResult(xml);
    }
});
```

Ce code permet d'instancier la classe responsable de la transformation du fichier XML en lui passant en paramètre l'identifiant de la balise où devront être affichés les résultats.

Modélisation de ce fichier et de son schéma d'ensemble :

Figure 14

Représentation du fichier jTransRequest.js

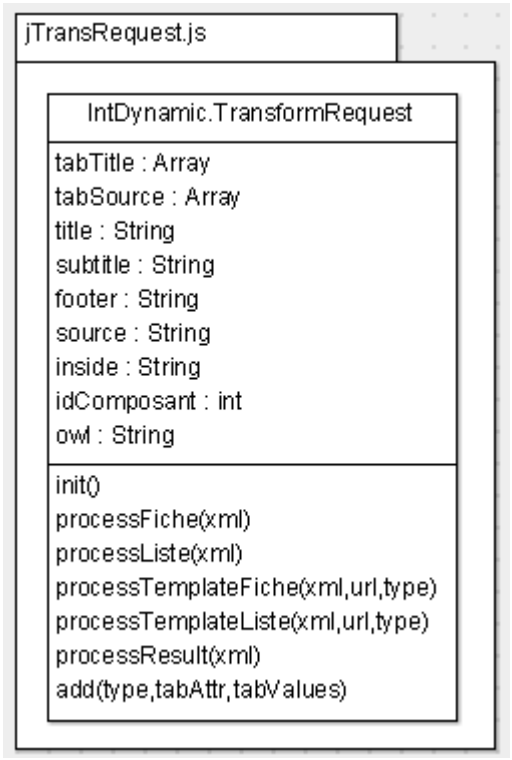
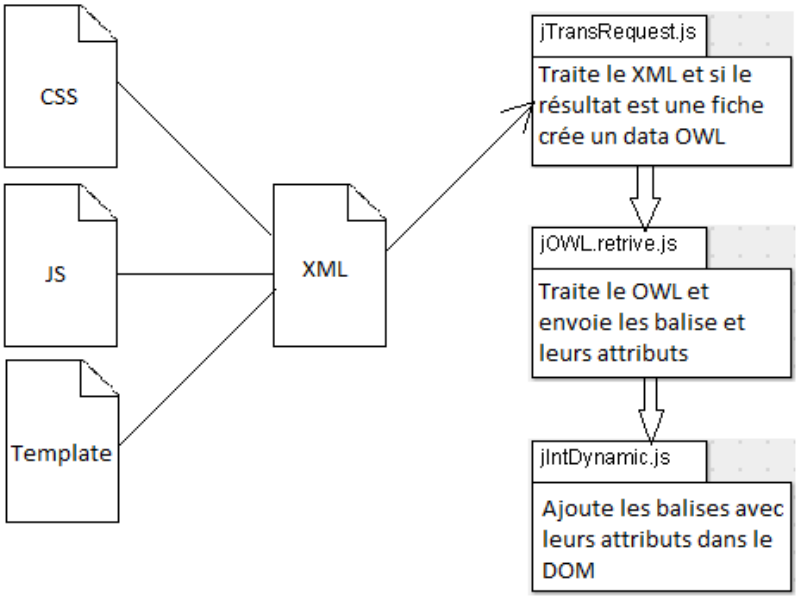


Figure 15

Représentation de l'ensemble des fichiers



6. Technologies

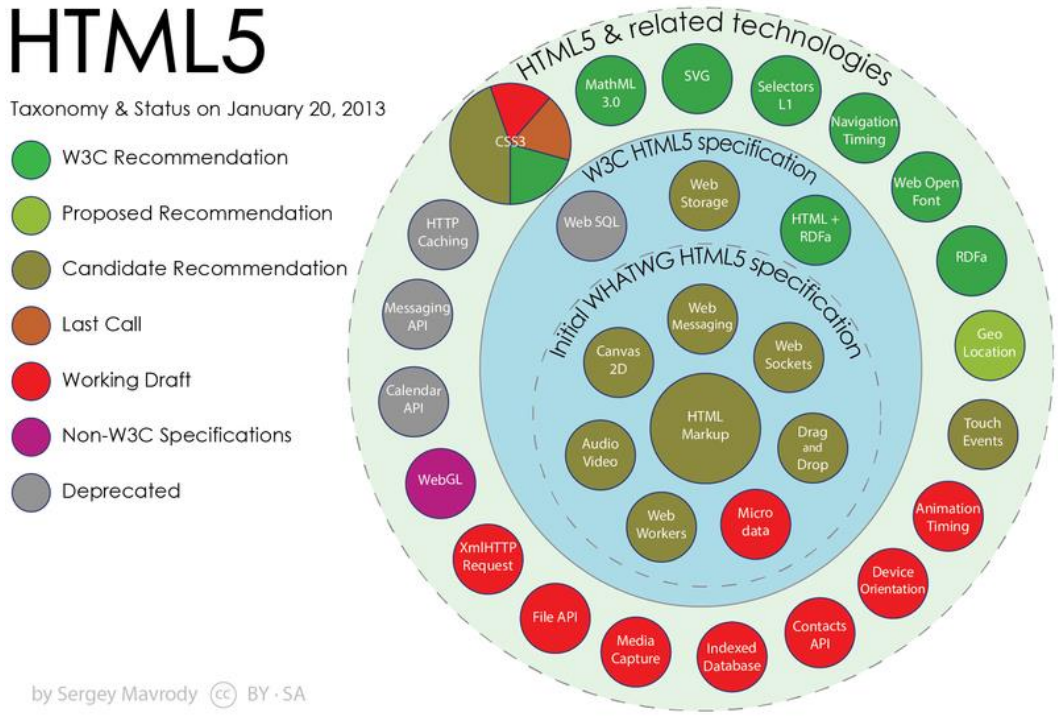
6.1 ***HTML 5***

HTML 5 va probablement devenir le nouveau standard HTML. HTML 5 est encore en cours de spécification et n'est pour le moment pas encore supporté entièrement par tous les navigateurs.[24]

HTML 5 introduit plusieurs nouveautés comme les balises <video> ou <audio> qui permettent de prendre en charge le multimédia ou la balise <canvas> qui permet de créer un espace interactif à l'image de flash.

Figure 16

Représentation des technologies HTML5



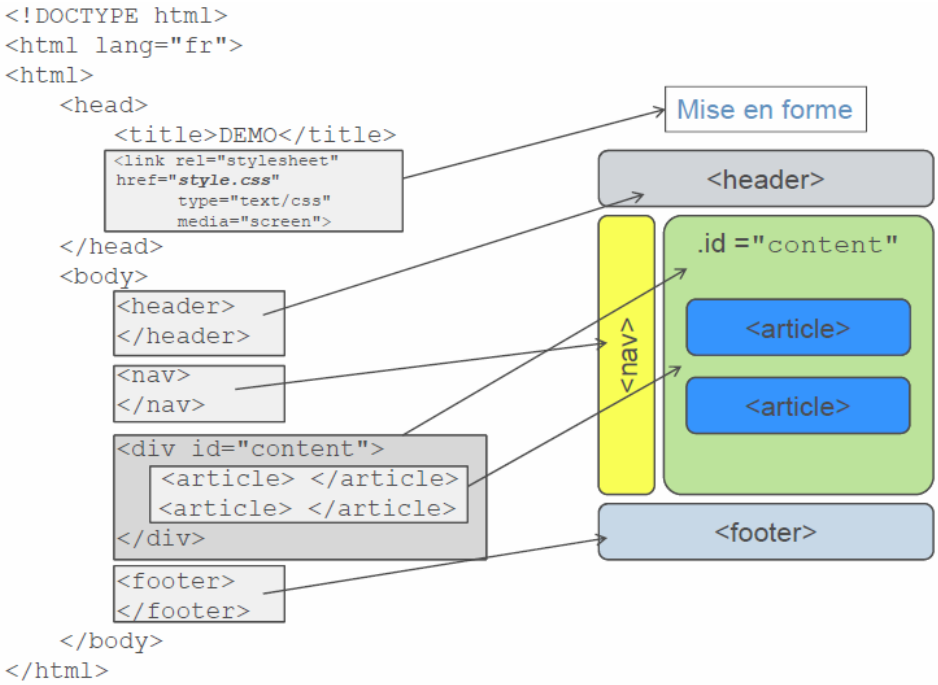
Source : Sergey Mavrody, via Wikimedia Commons

Les nouveautés pertinentes dans le cadre du projet sont la possibilité d'ajouter des vidéos et du son grâce aux nouvelles balises multimédias, la possibilité d'ajouter des nouvelles polices ou encore le déplacement (drag n drop) natif à l'aide de l'attribut « draggable ».

[24] Site internet listant les compatibilités par navigateurs : http://www.findmebyip.com/litmus/

Figure 17

Présentation d'une page simple en HTML 5



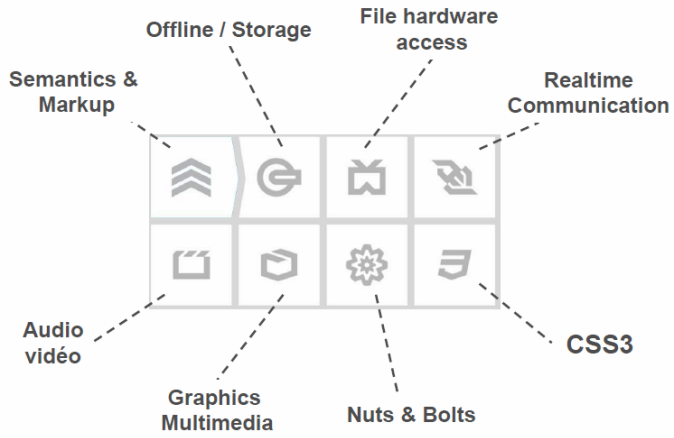
Source : http://www.w3schools.com/html/html5_intro.asp

http://www.alsacreations.com/article/lire/750-HTML5-nouveautes.html

La figure ci-dessus présente les nouveaux éléments de section de HTML5. Ceux-ci permettent de donner une sémantique aux éléments de la page web. Ce qui n'était pas le cas avec les balises telles la balise div ou encore p. La balise « header » par exemple permet d'introduire un en-tête de page. La balise « footer » à la même fonction que le header mais pour le bas de page. En conclusion, ces nouvelles balises permettent de mieux structurer la répartition des éléments sur la page.

Figure 18

Présentation des groupes fonctionnels HTML 5



Source : http://www.w3.org/html/logo/

La figure ci-dessus présente les principaux groupes fonctionnels HTML 5. « Semantics & Markup » permet de donner un sens à la structure de la page ainsi qu'à ses éléments. « Offline / Storage » permet aux applications web de travailler même sans connexion et le stockage local ainsi que la base de données autorisent la gestion des fichiers localement au niveau du navigateur.

« Audio vidéo » fait référence aux nouvelles balises d'HTML 5 qui permettent d'afficher un contenu vidéo ou audio. « Graphics Multimedia » permet d'afficher des objets 2D, 3D grâce à la balise canvas ou encore d'éditer des images (processing).

« CSS3 » permet entre autres, de créer des dégradés à la volée, d'afficher plusieurs images à la suite ce qui permet le découpage et l'ajout facile de cadres ou encore les arrondis d'images ou de cadres. Propriétés recherchées dans ce projet

6.2 JavaScript

«Le Javascript est un langage de script incorporé dans un document HTML. Historiquement il s'agit même du premier langage de script pour le Web. Ce langage est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes du côté client, c'est-à-dire au niveau du navigateur et non du serveur web.

Ainsi le langage Javascript est fortement dépendant du navigateur appelant la page web dans laquelle le script est incorporé, mais en contrepartie il ne nécessite pas de compilateur, contrairement au langage Java, avec lequel il a longtemps été confondu.

Javascript a été mis au point par Netscape en 1995. A l'origine, il se nommait LiveScript et était destiné à fournir un langage de script simple au navigateur Netscape Navigator

2. Il a à l'époque longtemps été critiqué pour son manque de sécurité, son développement peu poussé et l'absence de messages d'erreur explicites rendant dure son utilisation. Le 4 décembre 1995, suite à une association avec le constructeur Sun, Netscape rebaptise son langage Javascript (un clin d'oeil au langage Java développé par Sun). A la même époque, Microsoft mit au point le langage Jscript, un langage de script très similaire. Ainsi, pour éviter des dérives de part et d'autre, un standard a été défini pour normaliser les langages de script, il s'agit de l'ECMA 262, créé par l'organisation du même nom (ECMA, European Computer Manufactures Association).»[25]

6.2.1 Jquery

« jQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax[26]) et HTML, et a pour but de simplifier des commandes communes de JavaScript. La première version date de janvier 2006. » [27]

jQuery est très léger(32kB), ce qui permet de limiter la bande passante, il peut s'appuyer sur CSS3 conjointement avec les opérations sur le DOM et est compatible avec la plupart des navigateurs. Il peut être importé depuis le CDN de Google. De plus, si un autre site visité préalablement a fait un accès au CDN, il est fort probable que le plugin soit déjà en cache, ce qui permet une rapidité accrue.

L'appel d'une page par Ajax avec jQuery est facilité, il faut simplement lui donner une URL, les données nécessaires et le comportement à réaliser en cas de succès.

Par exemple :

```
$.ajax( {
    type: "GET",
    url: "./data/resultFiche.xml",
    dataType: "xml",
    success: function(xml) {
        monOwl = new
              IntDynamic.TransformRequest("interfaceDynamic");
        monOwl.processResult(xml);
    }
});
```

[25] Source : http://www.commentcamarche.net/contents/javascript/jsintro.php3

[26] Ajax envoie une requête asynchrone ce qui permet de ne pas devoir attendre la réponse envoyée par le serveur web et l'on peut ainsi continuer à travailler.

[27] Wikipédia de jQuery : http://fr.wikipedia.org/wiki/JQuery

Ce code permet de charger le fichier XML « resultFiche » et une fois le chargement terminé, d'instancier la classe « IntDynamic.TransformRequest ». Le comportement traite le résultat XML et le présente sous la forme d'une fiche ou d'un tableau.

6.3 RDF/OWL

6.3.1 RDF

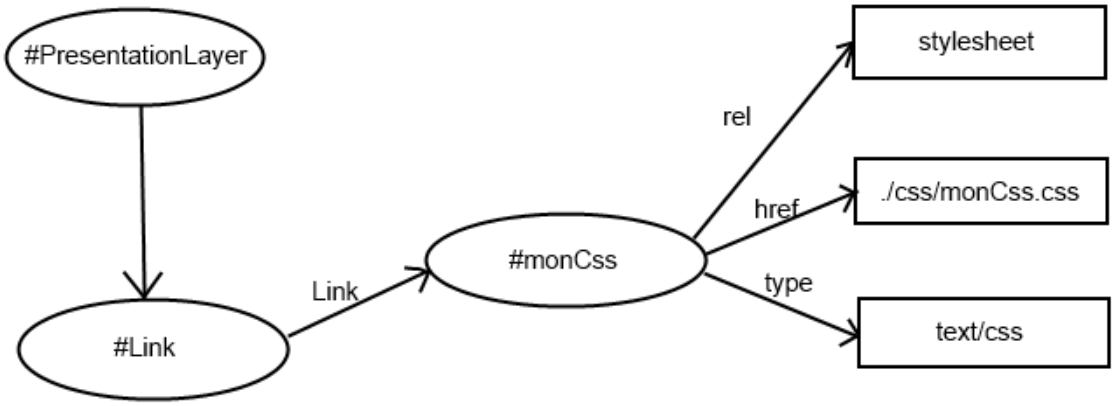
RDF est le diminutif pour « Resource Description Framework » il est destiné à donner un sens aux ressources web sous forme de triplets « (sujet, prédicat, objet) » cela permet de décrire les objets qui sont en relations et la qualité de leurs relations.

- Le sujet représente la ressource à décrire ;
- Le prédicat représente un type de propriété applicable à cette ressource ;
- L'objet représente une donnée ou une autre ressource : c'est la valeur de la propriété[28].

Un exemple de RDF provenant de mon code :

Figure 19

Graphique représentant un exemple de mon fichier RDF



Le code RDF que le graphique représente :

```
<Link rdf:about="#monCss">
      <rdf:type rdf:resource="#Link"/>
      <rel>stylesheet</rel>
      <href>./css/monCss.css</href>
      <type>text/css</type>
</Link>
```

[28] Page Wikipédia : http://fr.wikipedia.org/wiki/Resource_Description_Framework

Cela nous permet donc de vérifier qu'il est très facile de créer des objets, des relations et des attributs en RDF.

6.3.2 RDFS

« RDF Schema ou RDFS est un langage extensible de représentation des connaissances. Il appartient à la famille des langages du Web sémantique publiés par le W3C. RDFS fournit des éléments de bases pour la définition d'ontologies ou vocabulaires destinés à structurer des ressources RDF. Avec ces ressources structurées avec RDFS dans un triplestore, il est possible d'utiliser le langage de requête SPARQL pour les atteindre à travers le Web.

La première version de RDFS a été proposée en mars 1999, et la recommandation finale publiée par le W3C en février 2004. Les composants principaux de RDFS sont intégrés dans un langage d'ontologie plus expressif, OWL. » [29]

Un exemple de Schéma RDF est FOAF[30] (Friend of a Friend), il permet de définir des personnes et leurs relations comme par exemple leur nom, titre, pseudonyme, page web ou encore téléphone. Une norme de notation des ontologies consiste à commencer par une majuscule pour les classes et une minuscule pour les propriétés.

6.3.3 OWL

OWL est l'abréviation de Web Ontology Language et est construite sur le modèle de données RDF/XML.

Ce texte tiré d'un article de l'EPFL[31] explique les couches de base de web sémantique :

« XML, RDF et OWL constituent les trois couches de base du Web Sémantique : XML est le support de sérialisation sur lequel s'appuient RDF et OWL pour définir des structures de données et les relations logiques qui les lient. »[32]

Ce texte est suivi d'un schéma des couches :

[29] Page Wikipédia : http://fr.wikipedia.org/wiki/RDFS

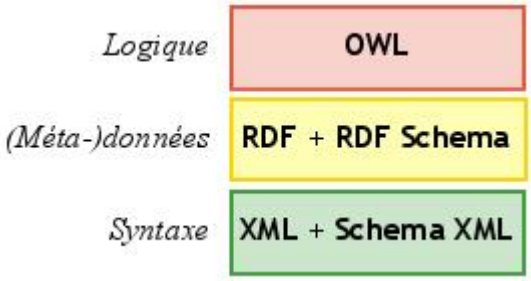
[30] Site officiel de FOAF : http://www.foaf-project.org/

[31] Article de l'EPFL sur l'OWL : http://flashinformatique.epfl.ch/spip.php?article1182

[32] Page Wikipédia : http://fr.wikipedia.org/wiki/Web_Ontology_Language

Figure 20

Schéma représentant les couches du web sémantique



Source : http://flashinformatique.epfl.ch/spip.php?article1182

Un exemple d'un header OWL pris de mon code :

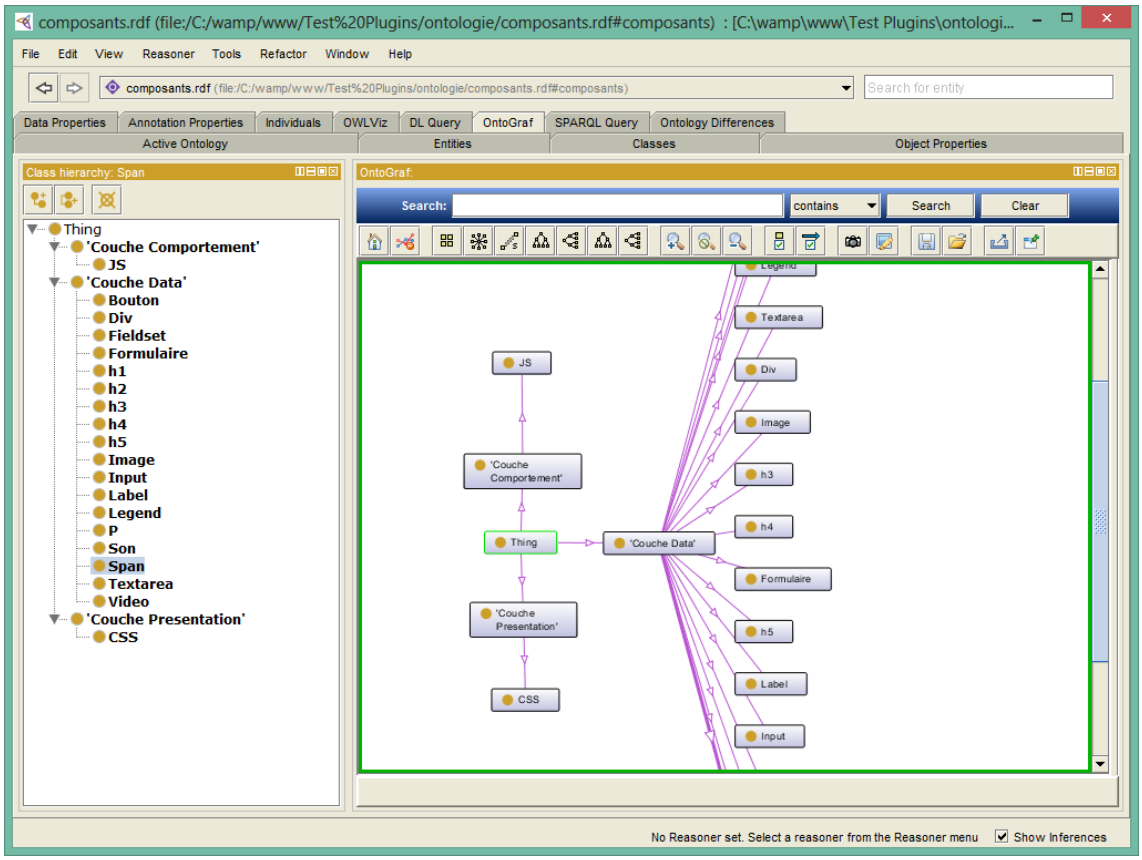
```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >]>
<rdf:RDF
        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:owl="http://www.w3.org/2002/07/owl#">
        <owl:Ontology rdf:about="#composants">
                <rdfs:comment>Composants exemple</rdfs:comment>
                <rdfs:label>Composants exemple</rdfs:label>
        </owl:Ontology>
…
</rdf:RDF>
```

6.3.4 Logiciel d'édition d'ontologies « Protégé »

Protégé est un logiciel open source sous la licence « Mozilla Public License »[33] et permet l'édition et la création d'ontologies. Les ontologies produites par protégé peuvent être exportées dans divers formats comme RDF(S), OWL ou XML. Ce logiciel est basé sur la plate-forme Java et requière donc un JRE pour fonctionner. Une version du logiciel permet d'éditer des ontologies depuis un navigateur, mais cette fonctionnalité en est encore au stade de développement.

Figure 21

Interface de Protégé 4.2



[33] Détail de la licence Mozilla : http://www.mozilla.org/MPL/

6.4 SPARQL

« RDF est un format de données de graphe orienté et étiqueté pour représenter des informations dans le Web. Cette spécification définit la syntaxe et la sémantique du langage d'interrogation SPARQL pour RDF. SPARQL peut être utilisé pour exprimer des interrogations à travers diverses sources de données, que les données soient stockées nativement comme RDF ou vues comme du RDF via un logiciel médiateur (middleware). SPARQL est capable de rechercher des motifs de graphe (graph patterns) obligatoires et optionnels ainsi que leurs conjonctions et leurs disjonctions. SPARQL gère également le test extensible des valeurs et la contrainte des interrogations par un graphe RDF source. Les résultats des interrogations SPARQL peuvent être des ensembles de résultats ou des graphes RDF.»[34]

Voici un exemple de requête SPARQL et son équivalent en SQL fait par mon collègue Olivier :

```
--------------------Requête SPARQL--------------------
SELECT ?machine ?fabricant ?groupe
WHERE {
   ?machine oli:fait_partie ?groupe.
   OPTIONAL { ?fabricant oli:fabrique ?machine }}

--------------------Requête SQL---------------------
SELECT machine, fabricant, groupe
FROM machine sujet, fait_partie predicat, groupe objet, fabricant sujet2,
fabrique predicat2
WHERE sujet.id = predicat.id_sujet
  AND objet.id = predicat.id_objet
  AND sujet2.id = predicat2.id_sujet
  AND objet.id = predicat2.id_objet;
```

6.5 SPARQL-DL

SPARQL-DL est une sous-catégorie distincte de SPARQL. C'est un moteur de requête implémenté dans l'API de OWL et est architecturé pour répondre à des questions spécifiques d'ontologies.

SPARQL-DL est supporté par le plugin de traitement de fichiers OWL : jOWL ce qui permet une recherche très rapide et efficace.

J'ai utilisé des requêtes SPARQL-DL dans le cadre de la gestion des fichiers OWL, car elles sont très rapides et permettent des résultats en temps presque réel.

[34] Source : http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/

Les types de requêtes SPARQL-DL que j'ai utilisés dans mon projet avec le plugin jOWL sont les suivantes :

Cette requête permet de récupérer tous les individus ainsi que leurs classes :

1. new jOWL.SPARQL_DL("Thing(?i), Type(?i, ?C)")

Cette requête permet de connaître la super classe :

2. new jOWL.SPARQL_DL("SubClassOf("+ that.constructor.maClasse +", ?C)")

Cette requête permet de connaître les propriétés et leur contenu de l'individu :

3. new jOWL.SPARQL_DL("PropertyValue("+ that.constructor.monIndividual +", ?p, ?c)")

Ces exemples permettent de voir la mise en œuvre de telles requêtes.

Conclusion

Les interfaces dynamiques IHM agiles sont un sujet très intéressant et cela pourrait permettre de développer de nouveaux types de sites Internet dont l'affichage du contenu ne serait plus figé à l'avance, mais évoluerait en fonction de la nature des données à présenter.

Personnellement, ce projet m'a permis d'approfondir mes connaissances HTML 5, CSS 3 et jQuery. J'ai pu découvrir des fonctions d'HTML 5 que je n'avais jamais utilisées auparavant. Ne connaissant que très peu les technologies RDF et OWL, il a fallu que je les étudie et les mettent en application.

Par rapport au planning ; celui-ci a été retardé, à cause de l'exploration et l'analyse du domaine qui fut plus long que prévu. La mise en place d'un prototype d'IHM agile recevant un fichier OWL a été ensuite plus facile à réaliser.

Les difficultés rencontrées durant le mandat ont été :

- la compréhension du domaine ;
- la première prise en main du plugin jOWL permettant le traitement des fichiers OWL ;
- la mise en œuvre du prototype de fichier OWL pour réaliser les tests de mon interface agile ;
- et pour finir, l'intégration des autres parties à l'interface.

Finalement, j'ai trouvé ce mandat très intéressant et passionnant et ce projet m'a permis d'accroître ma culture et ma compréhension des musées tout en découvrant des nouvelles technologies. Le support et les explications de M. **Johann Sievering** ont été très précieux et m'ont énormément aidé dans la mise en œuvre de mon mandat.

Lexique

CSS :	Feuilles de style en cascade
RDF :	Resource Description Framework
OWL :	Web Ontology Language
HTML :	Hypertext Markup Language
JS :	JavaScript
JQUERY :	Bibliothèque JavaScript libre
API :	Association pour le patrimoine industriel
IDE :	Integrated Development Environment
SPARQL :	SPARQL Protocol and RDF Query Language
URI :	Uniform Resource Identifier
IHM :	Interface Homme-machine
JRE :	Java runtime environment
GUI :	Graphical User Interface

Bibliographie

Ouvrages consultés :

CEDERHOLM, Dan. CSS3 for web designers. New York : A Book A part, 2010. 133p.

KEITH, Jeremy. HTML5 for web designers. New York : A Book A part, 2010. 96p.

ROQUES, Pascal. UML 2 : *Modéliser une application web* 4e édition. Paris : Eyrolles, 2008. 246p (Les Cahiers du Programmeur).

Sites internet consultés :

ASSOCIATION POUR LE PATRIMOINE INDUSTRIEL, Site de l'API [en ligne]. http://www.patrimoineindustriel.ch (consulté le 26 septembre 2012)

YOYO DESIGN. *RDF pour nous autres* [en ligne]. http://www.yoyodesign.org/doc/digital-web/rdf-for-the-rest-of-us/ (consulté le 8 novembre 2012)

ONTOLOGIEONLINE. *jOWL* [en ligne] http://jowl.ontologyonline.org/ (consulté du 22 novembre 2012 au 4 décembre 2012)

JAVASCRIPT MVC. *jQuery.Class* [en ligne] http://javascriptmvc.com/docs.html#!jQuery.Class (consulté le 26 novembre 2012)

SMASHING MAGASINE. *How to Use CSS3 Pseudo-Classes* [en ligne] http://coding.smashingmagazine.com/2011/03/30/how-to-use-css3-pseudo-classes/ (consulté le 3 décembre 2012)

WEBDESIGN TUTS+. *Bring Your Forms Up to Date With CSS3 and HTML5 Validation* [en ligne] http://webdesign.tutsplus.com/tutorials/site-elements/bring-your-forms-up-to-date-with-css3-and-html5-validation/ (consulté le 4 décembre 21012)

#CSS {débutant;}. *CSS 3 : déclarer une police de caractère non standard avec @font-face* [en ligne] http://css.mammouthland.net/css3/font-face.php (consulté le 5 décembre 2012)

TYMPANUS. *Accordion with CSS3* [en ligne] http://tympanus.net/Tutorials/CSS3Accordion/index3.html (consulté le 7 décembre 2012)

DINBROR. *Bpopup* [en ligne] http://dinbror.dk/bpopup/ (consulté le 7 décembre 2012)

Annexe 1
Aperçu de l'analyse du domaine

ACTEURS / RÔLES	TÂCHES / FONCTIONS	DROITS / PROFILS	BUT DU CAS D'UTILISATION	RÉSULTAT ATTENDU	IMPORTANCE
Récepteur	Faire une fiche quand il reçoit un objet	Il a le droit de modifier uniquement les champs indispensables à l'identification de l'objet	Quand-il reçoit un objet il veut pouvoir faire une fiche sommaire de celui-ci	Que la fiche soit ajoutée dans la base de données	2
Catalogueur	Faire une fiche approfondie en cours d'inventaire	Il a le droit de modifier tous les champs indispensables mais ne peut pas diffuser en externe et modifier une fiche en ligne	Quand-il fait l'inventaire il veut pouvoir faire des fiches approfondies des objets	Que la fiche soit ajoutée dans la base de données	2
Catalogueur	Faire une fiche approfondie en début d'inventaire	Il a le droit de modifier tous les champs indispensables mais ne peut pas diffuser en externe et modifier une fiche en ligne	Au début de l'inventaire il veut pouvoir faire des fiches approfondies des objets	Que la fiche soit ajoutée dans la base de données	1
Supercatalogueur	Validation des fiches en interne	Il a le droit de modifier tous les champs	Il veut pouvoir modifier et valider les fiches sur l'intranet	Que la fiche soit modifiée, validée dans l'intranet	2
Supercatalogueur	Validation des fiches sur le web	Il a le droit de modifier tous les champs / Il peut valider, diffuser ou supprimer une fiche sur internet	Il veut pouvoir modifier, supprimer et valider les fiches sur l'Internet	Que la fiche soit modifiée, validée ou supprimée sur Internet	2

Supercatalogueur	Faire une fiche approfondie en cours d'inventaire	Il a le droit de modifier tous les champs / Il peut extraire des images HD	Quand-il fait l'inventaire il veut pouvoir faire des fiches approfondies des objets	Que la fiche soit ajoutée dans la base de données	2
Supercatalogueur	Faire une fiche approfondie en début d'inventaire	Il a le droit de modifier tous les champs / Il peut extraire des images HD	Au début de l'inventaire il veut pouvoir faire des fiches approfondies des objets	Que la fiche soit ajoutée dans la base de données	1
Photographe	Créer des images ou autres médias en aval de la fiche	Déclenchement du processus de sauvegarde et de gravure	Il veut faire des photos ou autres médias d'un objet en aval de sa fiche	Que la photo ou média soit prise et sauvegardée	1
Photographe	Créer des images ou autres médias en amont de la fiche	Déclenchement du processus de sauvegarde et de gravure	Il veut faire des photos ou autre médias d'un objet en amont de sa fiche	Que la photo ou média soit prise et sauvegardée	1
Technicien de médias	Associer les médias aux fiches	Création de fiches très sommaires pour permettre l'enregistrement des images dans la base et leur association à cette fiche	Il veut créer une fiche sommaire pour enregistrer les images, médias dans la base de données et les associées à cette fiche	Que la photo ou média soit enregistrée dans la base et associé à la fiche crée	1
Technicien de médias	Associer les médias aux fiches	Accès exclusif à certaines fonctionnalités et certains champs techniques	Il veut associer les images, médias aux fiches et pouvoir rajouter certaines données techniques	Que les photos, médias soient ajoutés dans la base de données et associés à ces fiches	1
Administrateur technique	Attribuer des rôles aux membres du réseau	Accès à l'ensemble du système	Il veut pouvoir attribuer des rôles aux membres	Que les rôles soient bien attribués dans la base de données	2

Administrateur technique	Gérer les permissions sur les champs et les attribuer aux rôles	Accès à l'ensemble du système	Il veut pouvoir ajouter des permissions aux rôles	Que les permissions soient bien attribuées aux rôles dans la base de données	2
Administrateur technique	Pouvoir gérer les différentes bases et utiliser tous les outils des autres acteurs	Accès à l'ensemble du système	Il veut pouvoir gérer l'intégralité du système d'information	Que les différentes bases soient disponibles et que le CRUD se fasse sur la base de données	2
Administrateur technique	Pouvoir gérer les interfaces	Accès à l'ensemble du système	Il veut pouvoir gérer les diverses interfaces et leurs critères d'interrogation	Que les interfaces soient modifiées	2
Administrateur contenu	Corriger un ensemble de fiches	Accès à l'ensemble du système	Il veut pouvoir corriger un grand nombre de fiches à la fois	Que les fiches soient modifiées dans la base de données	3
Administrateur contenu	Ajout de nouveaux champs dans la base de données	Accès à l'ensemble du système	Il veut pouvoir ajouter des nouveaux champs dans la base de données	Que les nouveaux champs soient ajoutés dans la base de données et disponibles sur les fiches	3
Administrateur contenu	Modification de champs et leurs paramètres	Accès à l'ensemble du système	Il veut pouvoir modifier des champs et changer leurs paramètres	Que les champs soient modifiés dans la base de données et dans les fiches	3
Lecteur privilégié	Visualiser l'ensemble des champs même ceux qui sont confidentiels	Pas d'intervention sur la base / Visualisation de champs confidentiels	Il veut pouvoir visualiser la totalité des données	Que tous les champs lui soient affichés	2

Utilisateur	Visiter le site	Pas d'intervention sur la base	Il veut pouvoir visualiser des objets	Que l'objet soit affiché avec ses détails	2
Utilisateur	Recherche un objet en particulier	Pas d'intervention sur la base	Il veut pouvoir trouver l'objet qui l'intéresse	Que l'objet qu'il cherche soit trouvé	3
Utilisateur	Recherche un objet très particulier	Pas d'intervention sur la base	Il veut pouvoir trouver l'objet qui l'intéresse	Que l'objet qu'il cherche soit trouvé	3
Utilisateur	Imprimer la fiche en PDF	Pas d'intervention sur la base	Il veut pouvoir imprimer sa fiche au format PDF	Que le PDF soit créé	0
Imprimante PDF	Imprimer la fiche en PDF	Imprimante	Elle veut convertir la fiche en PDF	Qu'un PDF soit créé	0

Annexe 2
Courriel

Bonjour,

Je vous contacte, car je suis en train de finaliser mes diagrammes Use case métier et pour que ces diagrammes soient le plus fidèles au besoin métier que possible il me faudrait des informations complémentaires.

Je me permets donc de vous demander s'il vous était possible de répondre aux questions suivantes qui m'aideraient dans la mise en œuvre de mon travail et vous en remercie d'avance.

1. *Il me faudrait le détail des acteurs des musées/du projet*
    a. *Leur nom (qui est-il) (par ex : archiviste, photographe, conservateur, internaute)*
    b. *Leur fonction (ce qu'il fait) (par ex : photographe - il s'occupe de faire les photos et de les mettre sur le serveur)*
    c. *Leurs actions (ce qu'il peut faire) (par ex : internaute : Rechercher, Visualiser)*
    d. *Leur hiérarchie (de qui il dépend) (par ex : archiviste – réponds au directeur de musée)*
    e. *Leur périmètre (sur quoi il peut agir) (par ex : photographe – n'agit que sur la prise de photos et l'ajout de celles-ci)*
2. *Ensuite il me faudrait le détail des rôles*
    a. *Leur nom (par ex : Rechercher, Visualiser les œuvres …)*
    b. *Leurs actions (quoi) (par ex : Rechercher –> Permet de rechercher dans la base de données les collections)*
    c. *Leur domaine (sur quoi) (par ex : Rechercher -> base de donnée des musées)*
    d. *Leur prérequis (ce dont il a besoin pour travailler) (par ex : Imprimer a besoin du périphérique Imprimante)*

e. *Leur post-condition (résultat après action) (par ex : Imprimer en PDF -> fichier .PDF)*

f. *Leur temporalité (quand) (par ex : archiver ne se fait que 3 mois par année)*

Je vous remercie énormément d'avance de l'aide que vous allez m'apporter grâce à ces informations.

Grâce à vos informations, je vous renverrai une version complète du diagramme de Use Case que vous pourrez commenter au cas où il manquerait quoi que ce soit que vous trouveriez d'important.

Meilleures salutations,

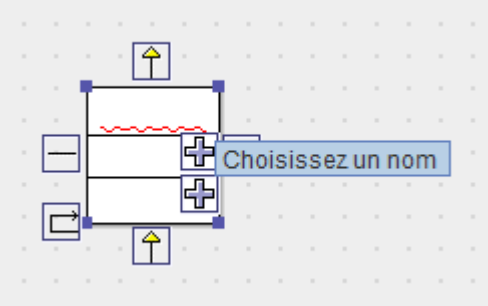
Michoud Loïc

Annexe 3
Explication de l'utilisation d'AgroUML

Ce logiciel est open source et dispose d'une licence « Eclipse Public Licence »[35]. Il est facile à prendre en main et est très complet. Pour ajouter un acteur ou une classe, il suffit de cliquer sur l'icône correspondante et ensuite de cliquer à l'endroit où l'on veut que l'objet apparaisse dans l'espace de travail. Un double clic suffit pour ajouter un nom, un attribut ou une fonction. Quand on clique une fois sur l'objet et que l'on passe le curseur sur celui-ci, des icônes s'affichent comme ci-dessous :

Figure 22

AgroUML ajout d'arguments



L'icône + sert à ajouter un attribut ou une fonction suivant l'endroit où l'on clique. Le – sert à ajouter une classe reliée à celle-ci. La flèche de généralisation du bas permet de rajouter une classe qui est sous-classe de la nôtre tandis que la flèche du haut ajoute une super classe et notre classe devient alors sous-classe.

Pour créer un nouveau diagramme, il suffit de cliquer sur l'item « Créer Diagramme » dans le menu et choisir le diagramme voulu. Les diagrammes possibles sont les suivants :

Diagramme de cas d'utilisation, Diagramme d'état, Diagramme de classes, Diagramme de séquence, Diagramme de collaboration, Diagramme d'activité et Diagramme de déploiement.

[35] Définition de la licence : http://www.eclipse.org/legal/epl-v10.html

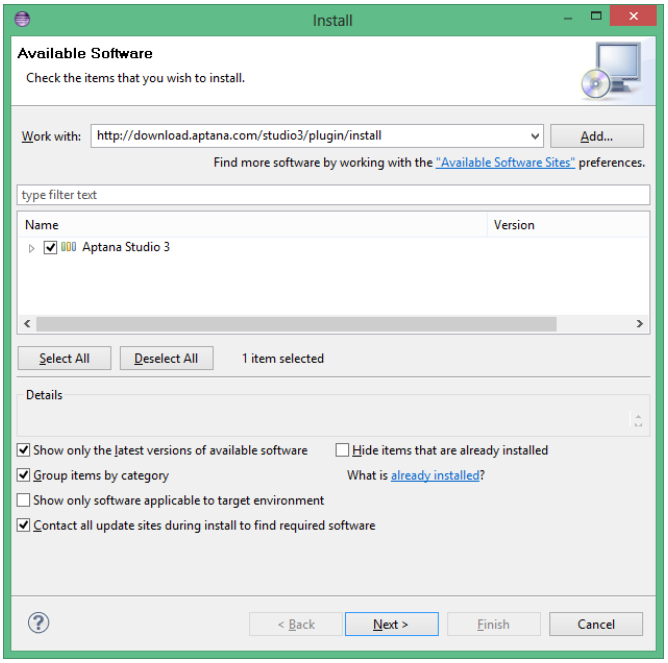
Annexe 4
Installation d'Eclipse et du plugin Aptana

Pour ce faire, l'installation d'Eclipse est requise. Celle-ci fonctionnant sous Java, elle a besoin de l'environnement de travail Java (Java runtime environment (JRE)). Celui-ci se trouve facilement sur le site officiel d'Oracle. Après avoir installé un JRE et avoir téléchargé une version d'Eclipse, il suffit de décompresser l'archive dans le dossier voulu et de lancer l'exécutable.

L'installation du plugin se fait très facilement, pour ce faire, il faut cliquer sur le menu « Help » et sélectionner l'option « Install New Software ».

Figure 23

Fenêtre Eclipse d'ajout de plugin

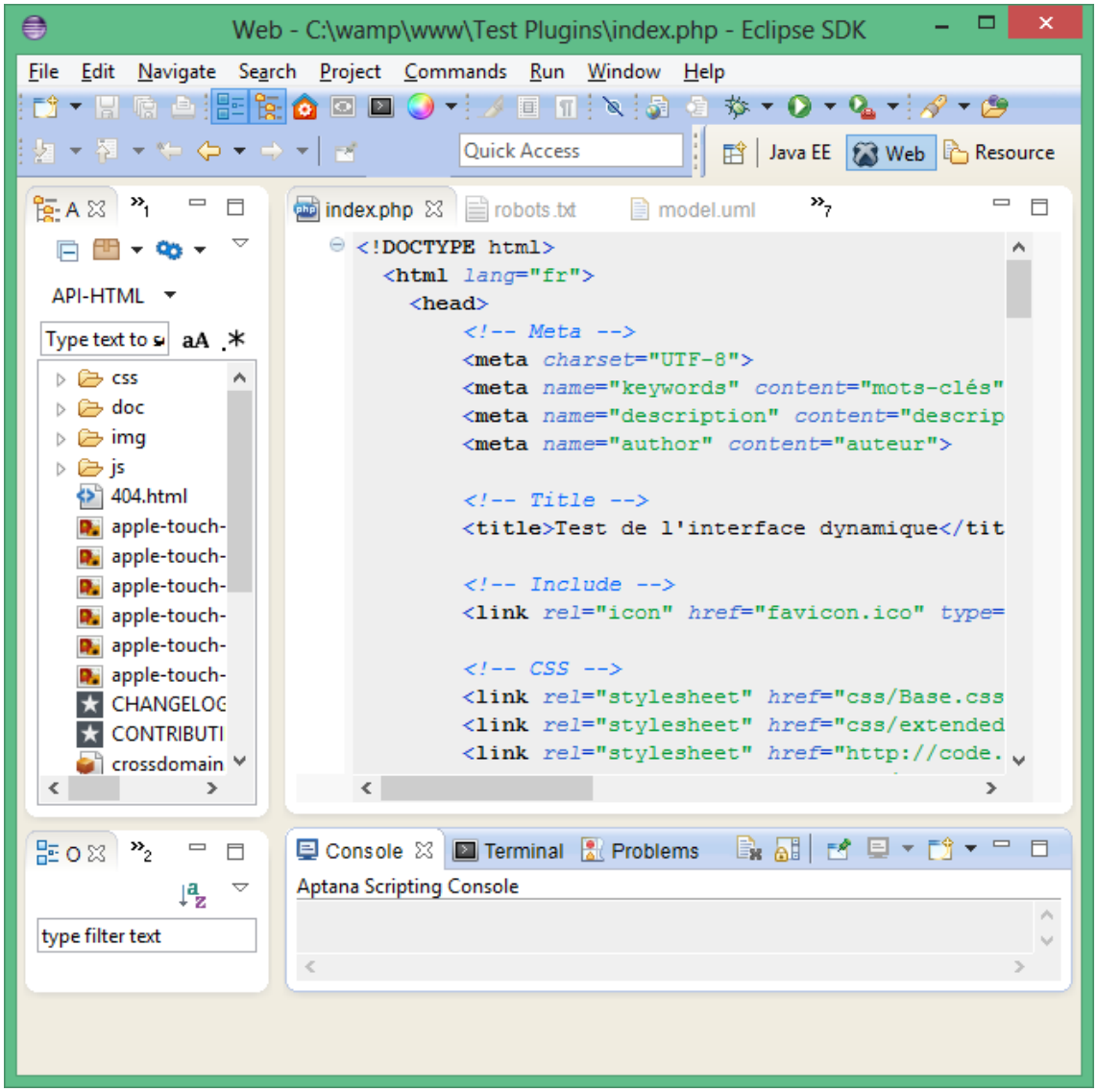


Comme dans la figure ci-dessus, il suffit d'insérer le lien du plugin dans la barre réservée à cet effet, le lien étant « http://download.aptana.com/studio3/plugin/install » puis d'appuyer sur la touche « Entrée » ou appuyer sur le bouton « Add ». Aptana Studio 3 apparaîtra alors dans la liste des plugins, il faudra ensuite le cocher et appuyer sur le bouton « Next ». La fenêtre suivante permettra d'afficher le détail de ce qui va être installé et pour poursuivre, il faudra appuyer sur le bouton « Next ». Une licence sera alors présentée qu'il faudra accepter pour pouvoir poursuivre l'installation en cliquant sur le bouton « Terminer ». Un redémarrage d'Eclipse sera alors nécessaire.

L'interface à disposition par le plugin ressemblera à cela :

Figure 24

GUI d'Eclipse avec le plugin Aptana



Pour commencer un nouveau projet, il faut appuyer sur le menu « File » -> « New » et choisir le type de projet voulu.

Annexe 5
Plugin permettant de transformer l'OWL en HTML 5

```
/*
    ### Fonction :
        Permet d'ajouter des éléments avec leurs attributs
        dans le header pour les scripts et css et dans
        l'élément ID rentré en paramètre.
    ### Plugin utilisé : jQuery, jQuery.class
    ### Auteur : Michoud Loïc - HEG
*/


/*
    Classe: Layer
    Espace de nommage: IntDynamic
    Fonction:
    @@
        Implémente les comportements
        de base des sous-classes.
    @@
    Sous-classes:
    @@
        PresentationLayer
        ComportementLayer
        DataLayer
    @@
*/
$.Class('IntDynamic.Layer',
/* @static */
{
},
/* @prototype */
{
    // Setup permet de simplifier l'initiation
    // car le setup est toujours appelé peut
    // importe quel init a été appelé.
    setup: function() {
        this.html = "";
        this.texte = "";
        this.lstAttr = new Array();
        this.lstAttrValue = new Array();
    },

    // Initiation de base sans arguments.
    init: function() {
    },

    // Initiation avec le nom de l'élément (ex: div)
    // et l'élément où cet élément sera inséré (ex: monContener).
    init: function( nomRoot, element ) {
        this.nomRoot = nomRoot.toLowerCase();
        this.element = element;
    },

    // Ajout d'un attribut à un array
    // en enlevant son préfixe
    // et ajout de la valeur de l'attribut
    // dans un autre array.
```

```
    options: function( attr, attrValue ) {
        this.lstAttr.push(attr);
        this.lstAttrValue.push(attrValue);
    },

    // Fonction ajout de base commun
    // à toutes les sous classes.
    add: function( ) {
        this.html =  $("<"+ this.nomRoot +" />").text(this.texte);

        for(key in this.lstAttr) {
            $(this.html).attr(this.lstAttr[key],
this.lstAttrValue[key]);
        }

        $(this.element).append(this.html);
    }
});

/*
    Classe: PresentationLayer
    Super-classe: Layer
    Espace de nommage: IntDynamic
    Fonction:
    @@
        Implémente les comportements
        spécifique à la couche de présentation.
    @@
*/
IntDynamic.Layer("IntDynamic.PresentationLayer",{
    add: function( ) {
        this._super();
    }
});

/*
    Classe: ComportementLayer
    Super-classe: Layer
    Espace de nommage: IntDynamic
    Fonction:
    @@
        Implémente les comportements
        spécifique à la couche de comportement.
    @@
*/
IntDynamic.Layer("IntDynamic.ComportementLayer",{
    add: function( ) {
        this._super();
    }
});

/*
    Classe: DataLayer
    Super-classe: Layer
    Espace de nommage: IntDynamic
    Fonction:
    @@
        Implémente les comportements
        spécifique à la couche de données.
    @@
*/
```

```
IntDynamic.Layer("IntDynamic.DataLayer",{
    add: function( ) {
        var monIndexText = this.lstAttr.indexOf("text");
        var monIndexInside = this.lstAttr.indexOf("inside");

        if(monIndexText >= 0){
            this.texte = this.lstAttrValue[monIndexText];
            this.lstAttr.splice(monIndexText, 1);
            this.lstAttrValue.splice(monIndexText, 1);
        }

        if(monIndexInside >= 0){
            this.element = this.lstAttrValue[monIndexInside];
            this.lstAttr.splice(monIndexInside, 1);
            this.lstAttrValue.splice(monIndexInside, 1);
        }

        this.element = "#" + this.element;

        this._super();
    }
});
```

Annexe 6
Prototype de fichier OWL

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:owl="http://www.w3.org/2002/07/owl#">

       <owl:Ontology rdf:about="#composants">
              <rdfs:comment>Composants exemple</rdfs:comment>
              <rdfs:label>Composants exemple</rdfs:label>
       </owl:Ontology>

       <!-- OWL Class Definition - Presentation Layer -->
       <owl:Class rdf:about="#PresentationLayer">
              <rdfs:label xml:lang="fr">Couche Presentation</rdfs:label>
              <rdfs:comment>La couche qui va s'occuper de la présentation (fichiers
CSS)</rdfs:comment>
       </owl:Class>

       <!-- OWL Class Definition - Comportement Layer -->
       <owl:Class rdf:about="#ComportementLayer">
              <rdfs:label xml:lang="fr">Couche Comportement</rdfs:label>
              <rdfs:comment>La couche qui va s'occuper du comportement (fichiers JS,
PHP, etc.)</rdfs:comment>
       </owl:Class>

       <!-- OWL Class Definition - Data Layer -->
       <owl:Class rdf:about="#DataLayer">
              <rdfs:label xml:lang="fr">Couche Data</rdfs:label>
              <rdfs:comment>La couche qui va s'occuper des données (composants
HTML)</rdfs:comment>
       </owl:Class>

       <!-- Couche Presentation Class Definition -->

              <!-- OWL Subclass Definition - Link -->
              <owl:Class rdf:about="#Link">
                     <rdfs:subClassOf rdf:resource="#PresentationLayer"/>
                     <rdfs:label xml:lang="fr">CSS</rdfs:label>
                     <rdfs:comment>Balise link pour mettre des fichiers
CSS</rdfs:comment>
              </owl:Class>

       <!-- Fin de la Couche Presentation Class Definition -->


       <!-- Couche Comportement Class Definition -->

              <!-- OWL Subclass Definition - Script -->
              <owl:Class rdf:about="#Script">
                     <rdfs:subClassOf rdf:resource="#ComportementLayer"/>
                     <rdfs:label xml:lang="fr">Balise script</rdfs:label>
                     <rdfs:comment>La balise script pour mettre des fichiers
javascripts</rdfs:comment>
              </owl:Class>

       <!-- Fin de la Couche Comportement Class Definition -->


       <!-- Couche Data Class Definition -->

              <!-- OWL Subclass Definition - Form -->
```

```
		<owl:Class rdf:about="#Form">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Formulaire</rdfs:label>
			<rdfs:comment>Un formulaire en html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Fieldset -->
		<owl:Class rdf:about="#Fieldset">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Fieldset</rdfs:label>
			<rdfs:comment>Un Fieldset en html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Legend -->
		<owl:Class rdf:about="#Legend">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Legend</rdfs:label>
			<rdfs:comment>Une legende pour notre fieldset en
html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - P -->
		<owl:Class rdf:about="#P">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">P</rdfs:label>
			<rdfs:comment>Un paragraphe en html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Span -->
		<owl:Class rdf:about="#Span">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Span</rdfs:label>
			<rdfs:comment>Un span en html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Div -->
		<owl:Class rdf:about="#Div">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Div</rdfs:label>
			<rdfs:comment>Un div en html</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Label -->
		<owl:Class rdf:about="#Label">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Label</rdfs:label>
			<rdfs:comment>Un label en html qui permet d'identifier des
inputs</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Image -->
		<owl:Class rdf:about="#Img">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Image</rdfs:label>
			<rdfs:comment>Une image</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - Video -->
		<owl:Class rdf:about="#Video">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">Video</rdfs:label>
			<rdfs:comment>Une vidéo</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - H1 -->
		<owl:Class rdf:about="#H1">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">h1</rdfs:label>
			<rdfs:comment>Un titre de grandeur H1</rdfs:comment>
		</owl:Class>

		<!-- OWL Subclass Definition - H2 -->
		<owl:Class rdf:about="#H2">
			<rdfs:subClassOf rdf:resource="#DataLayer"/>
			<rdfs:label xml:lang="fr">h2</rdfs:label>
			<rdfs:comment>Un titre de grandeur H2</rdfs:comment>
		</owl:Class>
```

```
        <!-- OWL Subclass Definition - H3 -->
        <owl:Class rdf:about="#H3">
                <rdfs:subClassOf rdf:resource="#DataLayer"/>
                <rdfs:label xml:lang="fr">h3</rdfs:label>
                <rdfs:comment>Un titre de grandeur H3</rdfs:comment>
        </owl:Class>

        <!-- OWL Subclass Definition - H4 -->
        <owl:Class rdf:about="#H4">
                <rdfs:subClassOf rdf:resource="#DataLayer"/>
                <rdfs:label xml:lang="fr">h4</rdfs:label>
                <rdfs:comment>Un titre de grandeur H4</rdfs:comment>
        </owl:Class>

        <!-- OWL Subclass Definition - H5 -->
        <owl:Class rdf:about="#H5">
                <rdfs:subClassOf rdf:resource="#DataLayer"/>
                <rdfs:label xml:lang="fr">h5</rdfs:label>
                <rdfs:comment>Un titre de grandeur H5</rdfs:comment>
        </owl:Class>

        <!-- OWL Subclass Definition - Button -->
        <owl:Class rdf:about="#Button">
                <rdfs:subClassOf rdf:resource="#DataLayer"/>
                <rdfs:label xml:lang="fr">Bouton</rdfs:label>
                <rdfs:comment>Un bouton en html 5</rdfs:comment>
        </owl:Class>

        <!-- OWL Subclass Definition - Input -->
        <owl:Class rdf:about="#Input">
                <!-- Shrubs is a subclassification of planttype -->
                <rdfs:subClassOf rdf:resource="#DataLayer"/>
                <rdfs:label xml:lang="fr">Input</rdfs:label>
                <rdfs:comment>Un input en html</rdfs:comment>
        </owl:Class>

<!-- Fin de la Couche Data Class Definition -->


<!-- Define the datatype property -->
<owl:DatatypeProperty rdf:ID="title"/>
<owl:DatatypeProperty rdf:ID="type"/>
<owl:DatatypeProperty rdf:ID="name"/>
<owl:DatatypeProperty rdf:ID="id"/>
<owl:DatatypeProperty rdf:ID="for"/>
<owl:DatatypeProperty rdf:ID="value"/>
<owl:DatatypeProperty rdf:ID="text"/>
<owl:DatatypeProperty rdf:ID="height"/>
<owl:DatatypeProperty rdf:ID="width"/>
<owl:DatatypeProperty rdf:ID="alt"/>
<owl:DatatypeProperty rdf:ID="checked"/>
<owl:DatatypeProperty rdf:ID="src"/>
<owl:DatatypeProperty rdf:ID="class"/>
<owl:DatatypeProperty rdf:ID="controls"/>
<owl:DatatypeProperty rdf:ID="form"/>
<owl:DatatypeProperty rdf:ID="action"/>
<owl:DatatypeProperty rdf:ID="method"/>
<owl:DatatypeProperty rdf:ID="formaction"/>
<owl:DatatypeProperty rdf:ID="formmethod"/>
<owl:DatatypeProperty rdf:ID="href"/>
<owl:DatatypeProperty rdf:ID="rel"/>
<owl:DatatypeProperty rdf:ID="pattern"/>
<owl:DatatypeProperty rdf:ID="placeholder"/>
<owl:DatatypeProperty rdf:ID="min"/>
<owl:DatatypeProperty rdf:ID="max"/>
<owl:DatatypeProperty rdf:ID="step"/>
<owl:DatatypeProperty rdf:ID="draggable"/>
<owl:DatatypeProperty rdf:ID="dropzone"/>
<owl:DatatypeProperty rdf:ID="required"/>
<owl:DatatypeProperty rdf:ID="subject"/>
<owl:DatatypeProperty rdf:ID="inside"/>


<!-- Couche Data Instances Definition -->

        <!-- Individual (Instance) Example RDF Statement -->
```

```
<H1 rdf:about="#monTitre">
        <rdf:type rdf:resource="#H1"/>

        <class>blue-gloss</class>
        <text>Mon formulaire</text>
</H1>


<Form rdf:about="#monForm">
        <rdf:type rdf:resource="#Form"/>

        <action>php/monPHP.php</action>
        <method>post</method>
        <id>monFormulaire</id>
</Form>

<P rdf:about="#pTitle">
        <rdf:type rdf:resource="#P"/>

        <class>glassText</class>
        <form>monFormulaire</form>
        <text>Voici un formulaire en HTML 5 !</text>
</P>

<P rdf:about="#pText">
        <rdf:type rdf:resource="#P"/>

        <id>pText</id>
        <form>monFormulaire</form>
</P>

<P rdf:about="#pUrl">
        <rdf:type rdf:resource="#P"/>

        <id>pUrl</id>
        <form>monFormulaire</form>
</P>

<P rdf:about="#pEmail">
        <rdf:type rdf:resource="#P"/>

        <id>pEmail</id>
        <form>monFormulaire</form>
</P>

<P rdf:about="#pNumeric">
        <rdf:type rdf:resource="#P"/>

        <id>pNumeric</id>
        <form>monFormulaire</form>
</P>

<P rdf:about="#pDate">
        <rdf:type rdf:resource="#P"/>

        <id>pDate</id>
        <form>monFormulaire</form>
</P>

<P rdf:about="#pSubmit">
        <rdf:type rdf:resource="#P"/>

        <id>pSubmit</id>
        <form>monFormulaire</form>
</P>

<Label rdf:about="#lText">
        <rdf:type rdf:resource="#Label"/>

        <form>monFormulaire</form>
        <for>iText</for>
        <inside>pText</inside>
        <text>Champ de texte avec placeholder</text>
</Label>

<Label rdf:about="#lUrl">
        <rdf:type rdf:resource="#Label"/>
```

```
        <form>monFormulaire</form>
        <for>iUrl</for>
        <inside>pUrl</inside>
        <text>Champ d'Url</text>
</Label>

<Label rdf:about="#lEmail">
        <rdf:type rdf:resource="#Label"/>

        <form>monFormulaire</form>
        <for>iEmail</for>
        <inside>pEmail</inside>
        <text>Champ Email</text>
</Label>

<Label rdf:about="#lNumeric">
        <rdf:type rdf:resource="#Label"/>

        <form>monFormulaire</form>
        <for>iNumeric</for>
        <inside>pNumeric</inside>
        <text>Format numerique</text>
</Label>

<Label rdf:about="#lDate">
        <rdf:type rdf:resource="#Label"/>

        <form>monFormulaire</form>
        <for>iDate</for>
        <inside>pDate</inside>
        <text>Format date</text>
</Label>

<Input rdf:about="#iText">
        <rdf:type rdf:resource="#Input"/>

        <id>iText</id>
        <name>monText</name>
        <form>monFormulaire</form>
        <type>text</type>
        <inside>pText</inside>
        <placeholder>Inserer votre texte ici</placeholder>
        <required>required</required>
</Input>

<Input rdf:about="#iUrl">
        <rdf:type rdf:resource="#Input"/>

        <id>iUrl</id>
        <name>monUrl</name>
        <form>monFormulaire</form>
        <type>url</type>
        <inside>pUrl</inside>
        <placeholder>http://www.devilryo.com</placeholder>
        <required>required</required>
</Input>

<Input rdf:about="#iEmail">
        <rdf:type rdf:resource="#Input"/>

        <id>iEmail</id>
        <name>monEmail</name>
        <form>monFormulaire</form>
        <type>email</type>
        <inside>pEmail</inside>
        <pattern>[^ @]*@[^ @]*</pattern>
        <placeholder>votre@dresse.ch</placeholder>
</Input>

<Input rdf:about="#iNumeric">
        <rdf:type rdf:resource="#Input"/>

        <id>iNumeric</id>
        <name>monNumeric</name>
        <form>monFormulaire</form>
        <type>number</type>
```

```
                        <inside>pNumeric</inside>
                        <value>5</value>
                </Input>

                <Input rdf:about="#iDate">
                        <rdf:type rdf:resource="#Input"/>

                        <id>iDate</id>
                        <name>maDate</name>
                        <form>monFormulaire</form>
                        <type>date</type>
                        <inside>pDate</inside>
                        <value>2012-03-01</value>
                </Input>

                <Span rdf:about="#sUrl">
                        <rdf:type rdf:resource="#Span"/>

                        <inside>pUrl</inside>
                        <class>form hint</class>
                        <text>Format correct "http://google.com"</text>
                </Span>

                <Span rdf:about="#sEmail">
                        <rdf:type rdf:resource="#Span"/>

                        <inside>pEmail</inside>
                        <class>form_hint</class>
                        <text>Format correct "name@something.com"</text>
                </Span>

                <Div rdf:about="#divBoutons">
                        <rdf:type rdf:resource="#Div"/>

                        <id>divBoutons</id>
                </Div>

                <Button rdf:about="#monButtonPHP">
                        <rdf:type rdf:resource="#Button"/>

                        <inside>divBoutons</inside>
                        <id>monButtonPHP</id>
                        <type>submit</type>
                        <form>monFormulaire</form>
                        <text>Soumettre le formulaire avec PHP !</text>
                </Button>

                <!-- Individual (Instance) Example RDF Statement -->
                <Button rdf:about="#monButtonJavascript">
                        <rdf:type rdf:resource="#Button"/>

                        <inside>divBoutons</inside>
                        <id>monButtonJavascript</id>
                        <type>button</type>
                        <form>monFormulaire</form>
                        <text>Bulle en JS !</text>
                </Button>

        <!-- Fin de la Couche Data Instances Definition -->


                <!-- Couche Comportement Instances Definition -->

                <Link rdf:about="#monCss">
                        <rdf:type rdf:resource="#Link"/>
                        <rdfs:comment>Le css de presentation de mes
composants</rdfs:comment>

                        <rel>stylesheet</rel>
                        <href>./css/monCss.css</href>
                        <type>text/css</type>
                </Link>

        <!-- Fin de la Couche Comportement Instances Definition -->


        <!-- Couche Presentation Instances Definition -->
```

```
		<script rdf:about="#monJs">
			<rdf:type rdf:resource="#script"/>
			<rdfs:comment>Un fichier javascript pour mes
boutons</rdfs:comment>

			<src>./scripts/monScript.js</src>
		</script>

	<!-- Fin de la Couche Presentation Instances Definition -->

</rdf:RDF>
```

Annexe 7
Code d'extraction du fichier OWL

```
/*
    ### Fonction :
        Permet de trouver les individuals dans le fichier OWL donné en paramètre,
        leurs classes et leurs super classes pour ensuite créer
        une instance de la bonne sous classe et l'ajouter à l'élément
        envoyé en paramètre.
    ### Plugin utilisé : jOWL, jQuery, jQuery.class
    ### Auteur : Michoud Loïc - HEG
*/


$.Class('RetriveDatasOwl',
/* @static */
{
    maClasse: "",
    monIndividual: "",
    maSuperClasse: "",
    options: {locale: 'fr'}
},
/* @prototype */
{
    init: function( nomFichier, nomDiv ) {
        this.nomOwl = nomFichier;
        this.appendDiv = nomDiv;
    },

    load: function( ) {
        var appendDiv = this.appendDiv;
        var appendHead = "head";
        var presentationLayer = "PresentationLayer";
        var comportementLayer = "ComportementLayer";
```

```
        var dataLayer = "DataLayer";
        var that = this;

        jOWL.parse(this.nomOwl);

        setTimeout(function(){//show individuals asynchronously

            new jOWL.SPARQL_DL("Thing(?i), Type(?i, ?C)")
                .execute(
                {   onComplete: function(results) {

                        arrIndividualsClasses = results.results;
                        arrIndividualsClasses.reverse();

                        for(key in arrIndividualsClasses) {
                            that.constructor.monIndividual = arrIndividualsClasses[key]['?i'].URI;
                            that.constructor.maClasse = arrIndividualsClasses[key]['?C'].URI;

                            new jOWL.SPARQL_DL("SubClassOf("+ that.constructor.maClasse +", ?C)")
                                .execute(
                                {   onComplete: function(results) {
                                        that.constructor.maSuperClasse = results.jOWLArray("?C");
                                    }
                                }
                            )

                            if(that.constructor.maSuperClasse == dataLayer) {
                                component = new IntDynamic.DataLayer(that.constructor.maClasse, appendDiv);
                            } else {
                                component = new IntDynamic.Layer(that.constructor.maClasse, appendHead);
                            }

                            new jOWL.SPARQL_DL("PropertyValue("+ that.constructor.monIndividual +", ?p, ?c)")
                                .execute(
                                {   onComplete: function(results) {
                                        arrProperty = results.jOWLArray("?p");
                                        arrPropertyResult = results.jOWLArray("?c");
```

```
                                        arrProperty.each(function(item, i) {
                                            component.options(item.URI, arrPropertyResult.get(i));
                                        })

                                        component.add();
                                    }
                                }
                            )
                        }
                    }
                )

            }, 200);
        }
});
```

Annexe 8
Fichier CSS de démonstration

```
#interfaceDynamic {
    display: inline-block;
    text-align: center;
    width: 100%;
    vertical-align: middle;
}

#divBoutons {
    margin: 0 auto;
    text-align: center;
}

#interfaceDynamic p {
    color: #000;
}

#interfaceDynamic form {
    display: inline-block;
    text-align: center;
    width: 100%;
}

#interfaceDynamic input, textarea {
    padding: 9px;
    border: solid 1px #E5E5E5;
    outline: 0;
    font: normal 13px/100% Verdana, Tahoma, sans-serif;
    width: 200px;
    background: -webkit-gradient(linear, left top, left 25,
from(#FFFFFF), color-stop(4%, #EEEEEE), to(#FFFFFF));
    background: -moz-linear-gradient(top, #FFFFFF, #EEEEEE 1px,
#FFFFFF 25px);
    box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
    -moz-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
    -webkit-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
}

#interfaceDynamic input:hover {
    padding: 9px;
    border: solid 1px #E5E5E5;
    outline: 0;
    font: normal 13px/100% Verdana, Tahoma, sans-serif;
    width: 200px;
    background: -webkit-gradient(linear, left top, left 25,
from(#FFFFFF), color-stop(4%, #EEEEEE), to(#FFFFFF));
    background: -moz-linear-gradient(top, #FFFFFF, #EEEEEE 1px,
#FFFFFF 25px);
    box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
    -moz-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
    -webkit-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
}

#interfaceDynamic input:hover, #interfaceDynamic textarea:hover,
#interfaceDynamic input:focus, #interfaceDynamic textarea:focus {
```

```
    border-color: #C9C9C9;
    -webkit-box-shadow: rgba(0, 0, 0, 0.15) 0px 0px 8px;
}

/* === HTML5 validation styles === */

#interfaceDynamic input:required, textarea:required {
    background: #fff url(../images/red_asterisk.png) no-repeat 98%
center;
}

#interfaceDynamic input:valid {
    background: #fff url(../images/valid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #5cd053;
    border-color: #28921f;
}

#interfaceDynamic input:required:valid, textarea:required:valid {
    background: #fff url(../images/valid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #5cd053;
    border-color: #28921f;
}

#interfaceDynamic input:focus:valid, textarea:focus:valid {
    background: #fff url(../images/valid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #5cd053;
    border-color: #28921f;
}

#interfaceDynamic input:focus:invalid, textarea:focus:invalid {
    background: #fff url(../images/invalid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #d45252;
    border-color: #b03535
}

#interfaceDynamic input:required:invalid, textarea:required:invalid {
    background: #fff url(../images/invalid.png) no-repeat 98% center;
    box-shadow: 0 0 5px #d45252;
    border-color: #b03535
}

/* === Form hints === */
#interfaceDynamic .form_hint {
    background: #d45252;
    border-radius: 3px 3px 3px 3px;
    color: white;
    margin-left:8px;
    padding: 1px 6px;
    z-index: 999; /* hints stay above all other elements */
    position: absolute; /* allows proper formatting if hint is two
lines */
    display: none;
}
#interfaceDynamic .form_hint::before {
    content: "\25C0";
    color:#d45252;
    position: absolute;
    top:1px;
    left:-6px;
}
```

```
#interfaceDynamic input:focus + .form_hint {display: inline;}
#interfaceDynamic input:required:valid + .form_hint {background:
#28921f;}
#interfaceDynamic input:required:valid + .form_hint::before
{color:#28921f;}

#interfaceDynamic textarea {
    width: 400px;
    max-width: 400px;
    height: 150px;
    line-height: 150%;
}

#interfaceDynamic label {
    font-family: SansationRegular;
    display: block;
    float: left;
    text-align: right;
    margin-left: 10px;
    color: #000;
    width: 200px;
}

.submit input {
    width: auto;
    padding: 9px 15px;
    background: #617798;
    border: 0;
    font-size: 14px;
    color: #FFFFFF;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
}
```

Annexe 9
Fichier JavaScript de démonstration

```
$(document).ready(function () {
    $("#monFormulaire").submit( function(e) {   // la soumission du
formulaire
        var myUrl = $("#monFormulaire").attr("action");

        e.preventDefault();
        dataString = $("#monFormulaire").serialize();

        callPHP(myUrl, dataString);
    });

    $('#monButtonJavascript').click(function(e){
        e.preventDefault();
        e.stopPropagation();
        callJS();
    });

    if (!Modernizr.inputtypes.number) {
        $('input[type = "number"]').spinner();
    }
    if (!Modernizr.inputtypes.date) {
        $('input[type = "date"]').datepicker();
    }
    $('input[type = "radio"]').buttonset();
    $('input[type = "checkbox"]').buttonset();
    $( "input[type=submit], button" ).button();
});

function callJS() {
    $('#popup').bPopup({
        contentContainer:'.content',
        loadUrl: 'html/test.html' //Uses jQuery.load()
    });
}

function callPHP(myUrl, myDatas) {
    $.post(myUrl, myDatas,
        function(data){
            $('#popup .content').html(data);
            $('#popup').bPopup();
        });
}
```

Annexe 10
Fichier HTML de démonstration

```
<!DOCTYPE html>
  <html lang="fr">
    <head>
              <!-- Meta -->
              <meta charset="UTF-8">
              <meta name="keywords" content="mots-clés" />
              <meta name="description" content="description" />
              <meta name="author" content="auteur">

              <!-- Title -->
              <title>Inventaires systématiques et sémantiques du patrimoine et des musées : Interface homme-machine agile</title>

              <!-- Include -->
              <link rel="icon" href="favicon.ico" type="image/x-icon" />

              <!-- CSS -->
              <link rel="stylesheet" href="css/Base.css" type="text/css" />
              <link rel="stylesheet" href="css/extended.css" type="text/css" />
              <link rel="stylesheet" href="http://code.jquery.com/ui/1.9.2/themes/base/jquery-ui.css" type="text/css" />
              <link rel="stylesheet" href="css/sbreadcrumb.css" type="text/css" />
              <link rel="stylesheet" href="css/browser-detection.css" type="text/css" />
              <link rel="stylesheet" href="css/menu.css" type="text/css" />
              <link rel="stylesheet" href="css/accordion.css" type="text/css" />
              <link rel="stylesheet" href="css/icons.css" type="text/css" />
              <link rel="stylesheet" href="css/login.css" type="text/css" />
              <link rel="stylesheet" href="fonts/walkway/stylesheet.css" type="text/css" />
              <link rel="stylesheet" href="fonts/Sansation/stylesheet.css" type="text/css" />
              <link rel="stylesheet" href="css/jquery.dataTables_themeroller.css" type="text/css" />
              <link rel="stylesheet" href="css/ColVis.css" type="text/css" />

              <!-- JS -->
              <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>
              <script>window.jQuery || document.write('<script src="scripts/jquery-1.8.3.min.js"><\/script>')</script>
              <script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.10.0/jquery-ui.min.js"></script>
              <script src="scripts/browser-detection.js"></script>
              <script src="scripts/jquery.class.min.js"></script>
              <script src="scripts/jquery.bpopup-0.7.0.min.js"></script>
              <script src="scripts/modernizr.custom.58364.js"></script>
              <script src="scripts/jquery.dataTables.min.js"></script>
```

```
<script src="scripts/ColReorderWithResize.js"></script>
<script src="scripts/ColVis.min.js"></script>
<script src="scripts/jOWL.js"></script>
<script src="scripts/jOWL.retrive.js"></script>
<script src="scripts/jTransRequest.js"></script>
<script src="scripts/jIntDynamic.js"></script>

<script>
	$(document).ready(function() {
		//Exemple d'appel de la classe de transformation du XML
		/*
			$.ajax( {
				type: "GET",
				url: "./data/resultList.xml",
				dataType: "xml",
				success: function(xml) {
					monOwl = new IntDynamic.TransformRequest("interfaceDynamic");
					monOwl.processResult(xml);
				}
			});
		*/

		//Permet de déplacer les éléments
		$('.sortable').sortable();
		$('.image').tooltip();

		;(function($) {
			$(function() {

				//Ouvre une popup quand on clique sur le bouton login
				$('#btnLogin').on('click', function(e) {
					e.preventDefault();

					$('#popup').bPopup({
						contentContainer:'.content',
						loadUrl: 'html/login.html'
					});
				});

				//Quand on soumet notre recherche il envoie la recherche et traite la réponse
				$("#fRecherche").submit( function(e) {
					var maRecherche = $('#txtRecherche').val();

					e.preventDefault();

					//Envoie la recherche et traite la réponse
```

```
                                        //Les possibilités de recherche du fichier PHP sont:
                                        //fiche : Affiche une fiche
                                        //liste 1 : Affiche la liste n°1
                                        //liste 2 : Affiche la liste n°2
                                        //liste 1 et 2 : Affiche la liste n°1 et n°2
                                        //les possibilités ne sont pas sensible à la case.
                                        $.ajax( {
                                                type: "POST",
                                                data: $('#fRecherche').serialize(),
                                                url: "./php/recherche.php",
                                                success: function(reponse) {
                                                        var maReponse = reponse;
                                                        $('#interfaceDynamic').empty();
                                                        if(maReponse.indexOf("Recherche non trouvée") !== -1) {
                                                                $('#interfaceDynamic').append(reponse);
                                                        } else {
                                                                eval(reponse);
                                                        }
                                                }
                                        });

                                        /*
                                        $.ajax( {
                                                type: "POST",
                                                data: maRecherche,
                                                //Mettre l'url de l'agent
                                                url: "",
                                                dataType: "xml",
                                                success: function(xml) {
                                                        monOwl = new IntDynamic.TransformRequest("interfaceDynamic");
                                                        monOwl.processResult(xml);
                                                }
                                        });
                                        */
                                });
                        });

                });(jQuery);
        });
    </script>
</head>
    <body>
        <header id="hd1" class="blue-gloss">
            <hgroup>
                <h1>Inventaires systématiques et sémantiques du patrimoine et des musées</h1>
                <h2>Interface homme-machine agile</h2>
```

```
                </hgroup>
            </header>

            <div id="center">

                <!-- Element to pop up -->
                <div id="popup">
                    <span class="button bClose"></span>
                    <div class="content"></div>
                </div>

                <nav id="menu">
                    <div class="wrapper">
                        <div class="container">
                            <ul class="menu" rel="sam1">
                                <li class="active"><a href="#">Home</a></li>
                                <li><a href="#About">About</a></li>
                                <li><a href="#Search">Recherche</a></li>
                                <li><a href="#Rss Feed">RSS</a></li>
                                <li><a href="#Contact">Contact</a></li>
                                <li><a id="btnLogin" href="#Login">Login</a></li>
                            </ul>
                        </div>
                    </div>
                </nav>

                <div id="breadcrumb">
        </div>

                <div id="accordion">
                    <section class="ac-container">
                        <div>
                            <input id="ac-1" class="ot-container" name="accordion-1" type="checkbox" checked="true" />
                            <label for="ac-1" class="ot-container">Recherche</label>
                            <article id="recherche" class="ac-recherche-simple art-container">
                                <h1 class="blue-gloss">Recherche simple</h1>
                                <form id="fRecherche">
                                    <label for="txtRecherche">Rechercher :</label>
                                    <input id="txtRecherche" name="textRecherche" type="text" placeholder="Insérer le
contenu de votre recherche ici !" required />
                                    <button id="btnRecherche" type="submit">Lancer la recherche</button>
                                </form>
                            </article>
                        </div>
                        <div>
                            <input id="ac-2" class="ot-container" name="accordion-1" type="checkbox" />
```

```
                                        <label for="ac-2" class="ot-container">Resultats</label>
                                        <article id="resultats" class="ac-dynamic art-container">
                                                <p><div id="interfaceDynamic" class="sortable"></div></p>
                                        </article>
                                </div>
                                <div>
                                        <input id="ac-3" class="ot-container" name="accordion-1" type="checkbox" />
                                        <label for="ac-3" class="ot-container">Status</label>
                                        <article id="status" class="ac-small art-container">
                                                <p>
                                                        <div id="progressBar"></div>
                                                </p>
                                        </article>
                                </div>
                                <div>
                                        <input id="ac-4" class="ot-container" name="accordion-1" type="checkbox" checked="true" />
                                        <label for="ac-4" class="ot-container">Footer</label>
                                        <article class="ac-footer art-container">
                                                <footer id="footer">
                                                        <ul id="icons">
                                                                <li class="facebook"><a href="#non" title="Share on
Facebook">Facebook</a></li>
                                                                <li class="twitter"><a href="#non" title="Share on Twitter">Twitter</a></li>
                                                                <li class="rss"><a href="#non" title="Subscribe to the RSS
feed">RSS</a></li>
                                                                <li class="linkedin"><a href="#non" title="Share on
LinkedIn">LinkedIn</a></li>
                                                                <li class="google"><a href="#non" title="Bookmark with
Google">Google</a></li>
                                                        </ul>
                                                </footer>
                                        </article>
                                </div>
                        </section>
                </div>

        </div>
    </body>
```